COMPUTADORAS

ELECTRONICAS

Prólogo

En la unidad 0 se hace una introducción, definiendo algunos conceptos previos.

En la unidad 1 se muestran las diversas tecnologías en las que se fueron construyendo las computadoras.
Luego se hace una breve revisión de los principios de circuitos digitales secuenciales y combinacionales, para comprender luego el Hardware principal de la computadora, como ser la Unidad Lógico Aritmética y la Unidad De Memoria. También se exponen las principales tecnologías de semiconductor con las que son construidas las computadoras actuales. Y se hace una descripción funcional de todas y cada una de las partes elementales de la computadora.
Finalmente se describen los dos tipos básicos de arquitectura de computadoras ejemplificando cada uno.

En la unidad 2 se hace un repaso de sistemas posicionales de numeración haciendo especial énfasis en el hexadecimal y binario. Además se desarrollan los diferentes códigos binarios con los que la maquina almacena y procesa la diversa información digital.

En la unidad 3 se hace una integración de las unidades anteriores en la programación de un microcontrolador específico. Haciendo uso de un entorno de desarrollo integrado para tal fin. Utilizando el lenguaje Ensamblador ya que al ser de bajo nivel se puede tener una compresión mas profunda de lo que ocurre a nivel Hardware y poder relacionarlo con el Software.

En la unidad 4 finalmente se expone el lenguaje C, luego de haber pasado por el Ensamblador se busca un mayor nivel de abstracción para poder resolver problemas de mayor complejidad.

Este material se realizó con fines didácticos y se encuentra en proceso de desarrollo. El mismo puede estar sujeto a errores y/o contenido que debería ampliarse y/o temas redundantes.

Yamil Saiegh

UNIDAD Nº 0

- Sistema Analógico y Digital.
- Hardware y Software.
- Abstracción.

UNIDAD Nº 1

Generaciones De Computadoras Electrónicas
1º: Válvula
2º: Transistor
3º: Circuito Integrado

- **Compuertas Lógicas**
  - Expresión algebraica
  - Simbología
  - Tabla de verdad

- **Circuitos Lógicos**
  - Combinacionales (sin memoria)
  - Secuenciales (con memoria)

Familias Lógicas TTL y CMOS
Características eléctricas:
- Potencia
- Velocidad
- Tiempos de propagación
- Tensiones

Descripción Funcional De Las Partes De La Computadora
- Unidad De Memoria
  - Clasificaciones.
  - Tecnologías.

- Unidad Central De Procesamiento (CPU)
  - Unidad De Control (UC).
  - Unidad Lógico Aritmética (ALU).

- Unidad De Entrada Y Salida
  - Sistema Básico de Entrada/Salida (BIOS).
  - Puerto Paralelo.
  - Puerto Serie - Transmisor-Receptor Asíncrono Universal (UART).
  - Periféricos.
  - Conversores analógico-digital y digital-analógico.

Arquitectura
- Harvard
- Von Neumann

UNIDAD Nº 2

Sistemas Posicionales De Numeración
- Hexadecimal
- Decimal
- Octal
- Binario

Conversiones entre sistemas

Códigos Binarios Para La Representación Y Procesamiento De La Información

Representación De Números
- Naturales: Natural
- Enteros: Signo y magnitud, Complemento a 1 y complemento a 2
- Reales: Coma fija y Coma flotante

Operaciones aritméticas

Representación De Caracteres
- ASCII
- ASCII Extendido

Códigos De Despliegue
Código De Siete segmentos

Códigos De Instrucciones
Código Máquina

UNIDAD Nº 3

Lenguaje Ensamblador
Representación mediante diagramas de flujo.

- Tipos de Datos y Declaraciones
- Control del Flujo
- Subrutinas
- Manejo de Puertos

Función y uso del:
- Editor
- Ensamblador
- Simulador
- Grabador

Aplicación a microcontroladores PIC.

UNIDAD Nº 4

Lenguaje C
Representación mediante diagramas de chapin.

- Tipos de Datos y Declaraciones
- Control del Flujo
- Vectores
- Funciones
- Manejo de Puertos

Función y uso del:
- Editor
- Compilador

Aplicación a microcontroladores PIC.

UNIDAD Nº0

Información Digital y Analógica

La información digital asume valores **discontinuos** dentro de un rango. Comúnmente asociada a al acción de contar y con los números enteros.

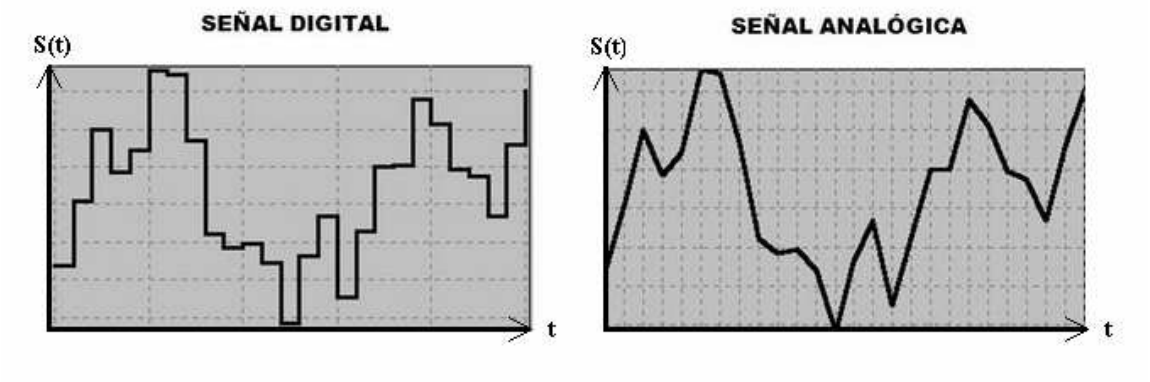
Ejemplo: Podemos contar 1, 2, 3 objetos, nunca 2,5 objetos.

La información analógica asume valores **continuos** dentro de un rango. Comúnmente asociada a la acción de medir y con los números reales.

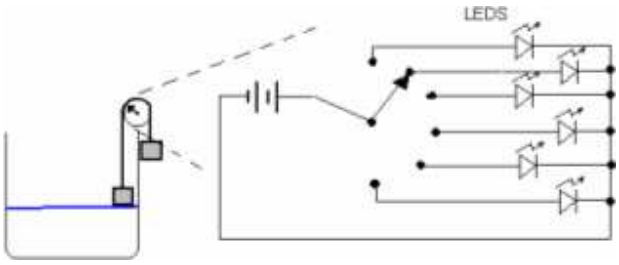
Ejemplo: Podemos medir una longitud de 5,32 metros.

Señal Digital y Analógica

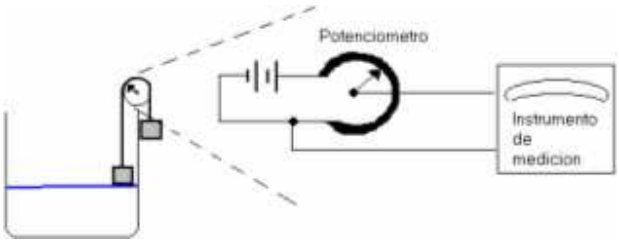
De lo dicho surge el aspecto que podría tener una señal digital y una señal analógica en función del tiempo.



Ejemplo: Sistema digital para informar el nivel de agua en un tanque

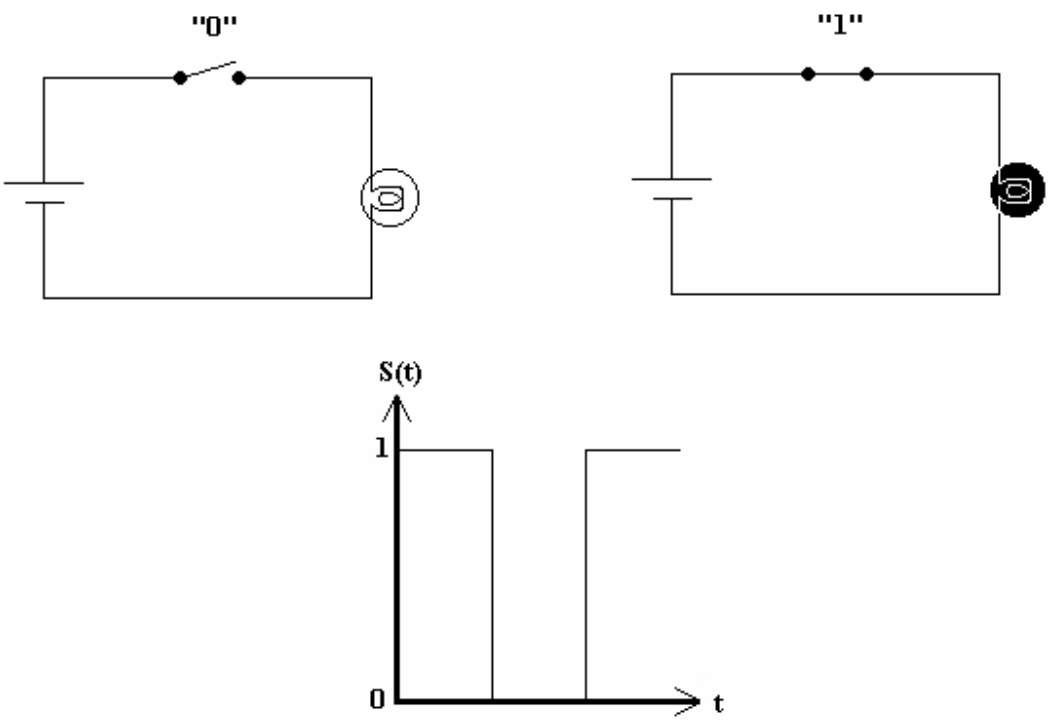


Ejemplo: Sistema analógico para informar el nivel de agua en un tanque



En esta asignatura nos interesará particularmente la información digital binaria, aquella que tiene únicamente dos estados discretos.

Ejemplo



Una lámpara o un interruptor son dispositivos que dan información digital binaria:

- El interruptor podrá estar abierto ó cerrado
- La lámpara podrá estar apagada ó encendida.

Asignándole arbitrariamente el símbolo "1" interruptor cerrado y lámpara encendida y el símbolo "0" interruptor abierto y lámpara apagada.
Cada combinación de llaves y lámparas queda simbolizada por una combinación de unos y ceros determinando una información digital binaria.

A la unidad de información digital binaria (0 y 1) se lo llama bit (binary unit)

Niveles Lógicos

Un bit podrá tener solo dos niveles lógicos posibles:

- Nivel alto (h) ó "1"
- Nivel bajo (l) ó "0"

Unidad	nibble	byte	word	Double word	Kilobyte	Megabyte	Gigabyte	Terabyte	Petabyte	Exabyte	Zettabyte	Yottabyte
Bits Agrupados	2^2	2^3	2^4	2^5	2^{13}	2^{23}	2^{33}	2^{43}	2^{53}	2^{63}	2^{73}	2^{83}

Programa: secuencia de instrucciones para resolver un problema.

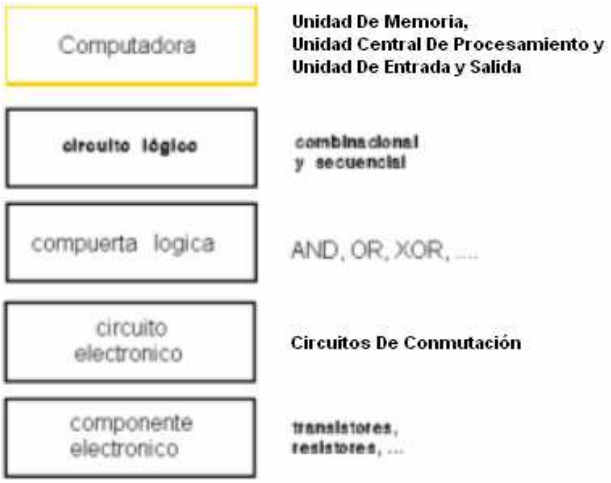
Máquina de propósito específico (Lógica Cableada)	**Máquina de propósito general (Lógica Programada)**
El programa es parte de la máquina.	El programa no es parte de la maquina.
Es necesario recablearla para resolver distintos tipos de problemas.	Solo es necesario reprogramarla para resolver distintos tipos de problemas.

Hardware (Objetos duros)	**Software (Objetos blandos)**
Elementos físicos que integran una computadora.	Programas que pueden ser ejecutados y datos que pueden ser procesados en el hardware de una computadora.

Abstracción

Construcción mental en la cual se aísla un elemento de su contexto o del resto de los elementos que lo acompañan, hace énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?".

En El Hardware



En El Software

Los lenguajes de programación se clasifican en niveles según si son más adecuados a la capacidad cognitiva humana (alto nivel) o a la capacidad ejecutora de las máquinas (bajo nivel).

Nivel Bajo

- Código máquina

Datos e instrucciones representadas en binario.

- Ensamblador

Datos e instrucciones representadas con palabras mnemotécnicas.

Ensamblador

Programa que permite traducir la codificación de un programa en lenguaje Ensamblador, a código máquina.

Nivel Alto

- C
- C++
- Java
- MATLAB
- Pascal

Compilador

Programa que permite traducir la codificación de un programa en lenguaje de alto nivel, a código máquina.

UNIDAD Nº1

Generaciones De Computadoras Electrónicas

Períodos en los que se divide la historia de las computadoras debido a la tecnología utilizada para crear el elemento lógico principal.

- **Primera generación (1945): Válvula Termiónica**

En 1884 Thomas Alva Edison al ver que con el uso el cristal de las lámparas incandescentes se iba oscureciendo, buscó formas de aminorar dicho efecto. Una de ellas fue la introducción en la ampolla de la lámpara de un electrodo en forma de placa, que se polarizaba eléctricamente con el fin de atraer las partículas que, al parecer, se desprendían del filamento. A esto lo patentó como "Efecto Edison". Este efecto es el principio de funcionamiento de las válvulas termiónicas.

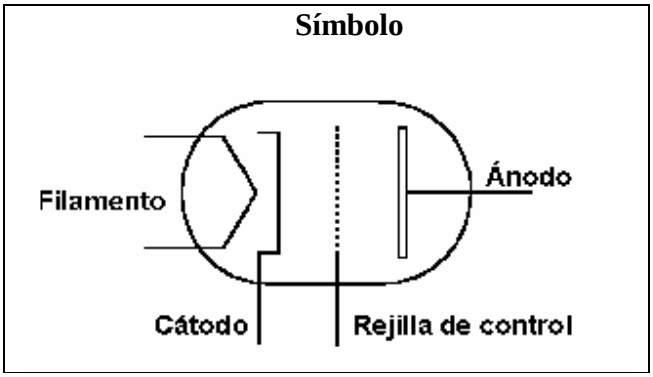
La válvula termoiónica de tres electrodos, o tríodo está constituida por una ampolla de vidrio a la que se le ha practicado el vacío y en la que se hallan encerrados tres electrodos:

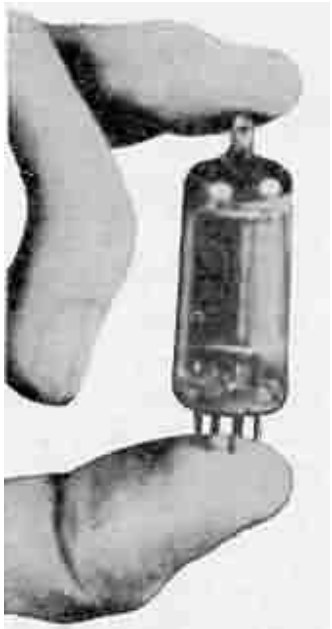
- El cátodo: filamento recubierto por una sustancia que se calienta mediante el paso de una corriente y desprende electrones, formándose una nube (nube termoiónica) por encima del mismo.

- El ánodo: placa metálica que rodea al cátodo a una cierta distancia y a la que se aplica un potencial positivo. Atrayendo de esta forma los electrones liberados por el cátodo. Dando lugar a una circulación de corriente electrónica.

- La rejilla: ubicada entre el ánodo y el cátodo. Variando una aplicación de tensión entre esta y el cátodo se pueden producir variaciones de la intensidad de corriente circulante entre cátodo y ánodo. De ahí la denominación de válvula.

Esto se utiliza para funciones como compuertas que dejan pasar la corriente o no, la base de la electrónica digital.





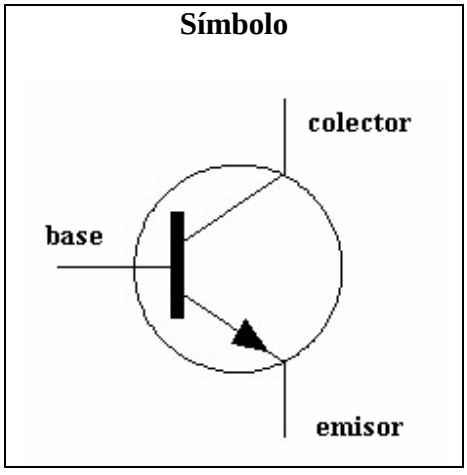
■ Segunda generación (1955): Transistor Bipolar

El transistor bipolar fue inventado por John Bardeen y Walter Brattain bajo la dirección de William Shockley en Bell Telephone Laboratories.

El transistor de unión bipolar es un dispositivo electrónico que permite controlar (a través de un terminal "base") el paso de la corriente a través de otros dos terminales (de "emisor" a "colector"). Cuando trabaja en la zona de corte y la de saturación se dice que trabaja en conmutación, como si fuera un interruptor. Esto es de gran aplicación en la electrónica digital.

Razones por las que el transistor reemplazó a la válvula termoiónica en la electrónica digital:

- Permitió la construcción de computadoras más poderosas, confiables, y menos costosas, ocupando menos espacio y produciendo menos calor que las que operaban a bases de válvulas termoiónicas.
- Las válvulas termoiónicas necesitan tensiones muy altas, del orden de las centenas de voltios, letales para el ser humano.
- Las válvulas consumen mucha energía, lo que las vuelve particularmente poco útiles para el uso con baterías.
- Son más pesadas. El chasis necesario para alojar las válvulas, los transformadores requeridos para suministrar la alta tensión.
- El tiempo medio entre fallas de las válvulas termoiónicas es muy corto comparado al del transistor.
- Rapidez de las operaciones.



- **Tercera generación (1965): Circuito Integrado**

El Circuito Integrado fue inventado por el ingeniero Jack Kilby en Texas Instrumets. Estaba hecho de germanio e integraba seis transistores.
Tienen bajo costo debido a que los CI son fabricados siendo impresos como una sola pieza por fotolitografía a partir de una oblea de silicio, permitiendo la producción en cadena de grandes cantidades con una tasa de defectos muy baja. Y Cuentan con un alto rendimiento debido a la miniaturización de todos sus componentes, el consumo de energía es considerablemente menor, a iguales condiciones de funcionamiento.

Nivel de integración		Año	Número aproximado de transistores
SSI	Small Scale Integration	1965	10<N<100
MSI	Medium Scale Integration	1968	100<N<1.000
LSI	Large Scale Integration	1972	1.000<N<10.000
VLSI	Very Large Scale Integration	1976	10.000<N<100.000
ULSI	Ultra Large Scale Integration	1980	100.000<N<1.000.000
GLSI	Giga Large Scale Integration	1989	1.000.000<N

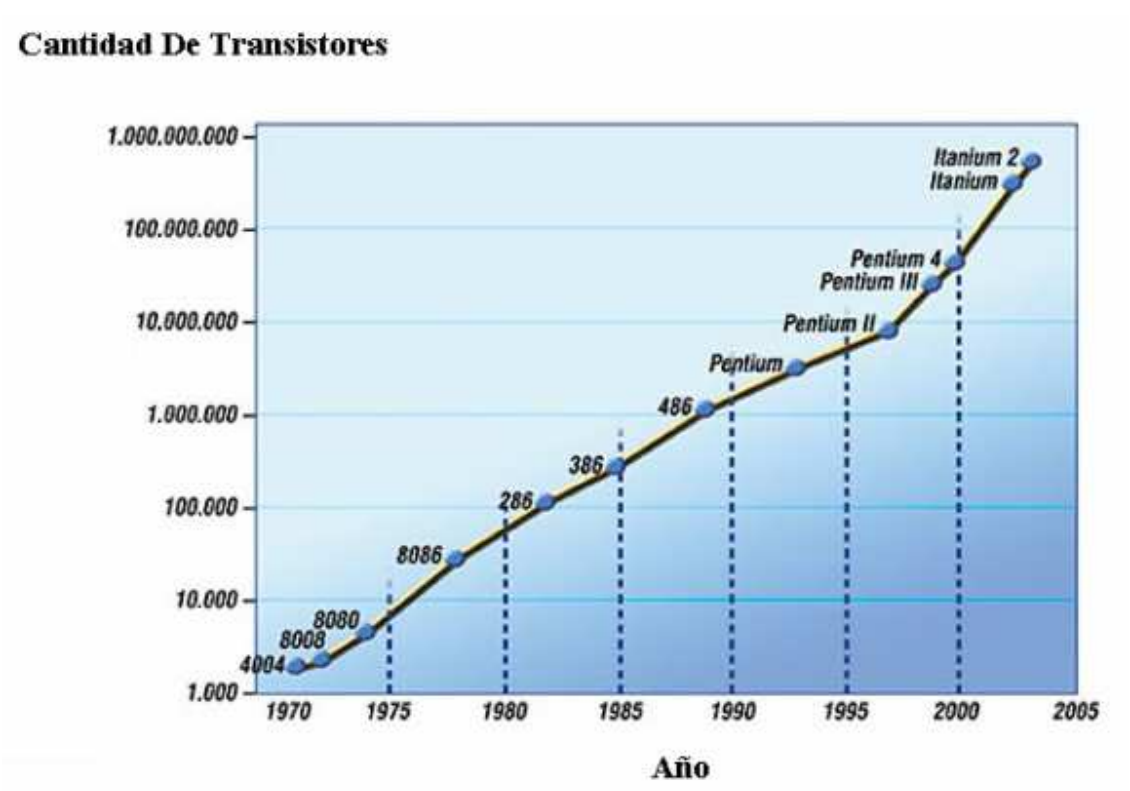
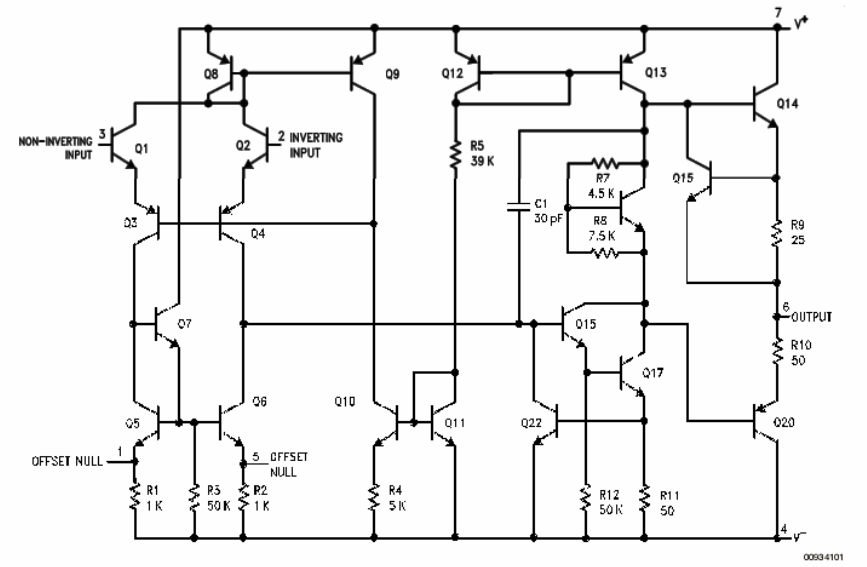
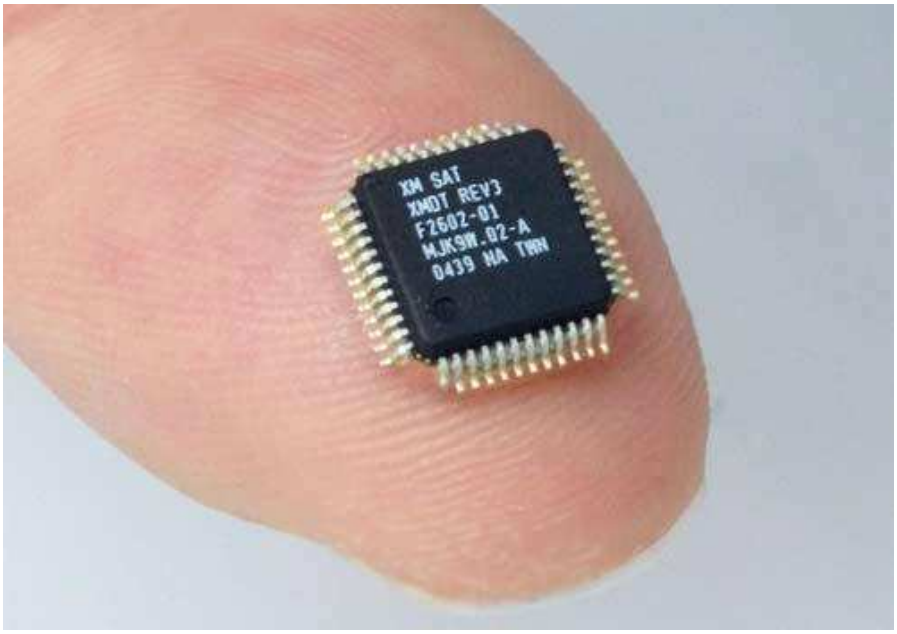


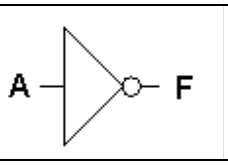
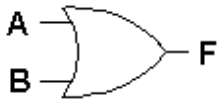
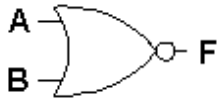
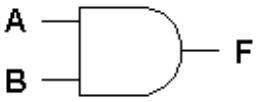
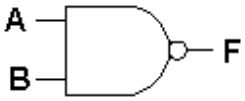
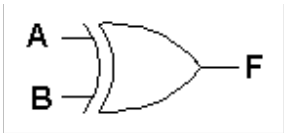
Diagrama Esquemático





Compuertas Lógicas

Dispositivo electrónico que opera según las expresiones del álgebra de Boole.

Nombre	Símbolo	Expresión Algebraica	Tabla De Verdad
NOT (NO)		$F = \overline{A}$	A | F 1 | 0 0 | 1
OR (O)		$F = A + B$	A | B | F 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1
NOR (NO O)		$F = \overline{A + B}$	A | B | F 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 0
AND (Y)		$F = A * B$	A | B | F 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
NAND (NO Y)		$F = \overline{A * B}$	A | B | F 0 | 0 | 1 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
XOR (O Exclusiva)		$F = (A + B) * (\overline{A} + \overline{B})$	A | B | F 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 0
XNOR (O Exclusiva Negada)		$F = \overline{(A + B) * (\overline{A} + \overline{B})}$	A | B | F 0 | 0 | 1 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1

Circuitos Lógicos

- Combinacionales (sin memoria)

El estado de las salidas en cada momento es una función lógica de la combinación presente en las entradas en ese momento.

- Secuenciales (con memoria)

El estado de las salidas en cada momento es una función lógica de la combinación presente en las entradas en ese momento y del estado anterior que tuvieron las salidas.

Tecnología Digital

Los circuitos integrados digitales se agrupan en "FAMILIAS LOGICAS" y cada una se diferencia en la tecnología del elemento principal utilizado. Los circuitos integrados de una misma "FAMILIA LOGICA" se pueden interconectar entre sí directamente.
Las más comunes son:

- TTL (Lógica Transistor Transistor)
  - Utiliza transistores bipolares.

- CMOS (Complentario De Semiconductores De Oxido Metálico)
  - Utiliza transistores de efecto de campo.

Tecnología	Serie
TTL	estándar
	de baja potencia
	shoottky
	shoottky de baja potencia
	shoottky avanzada
CMOS	estándar
	HC
	HCT

Parámetros Importantes

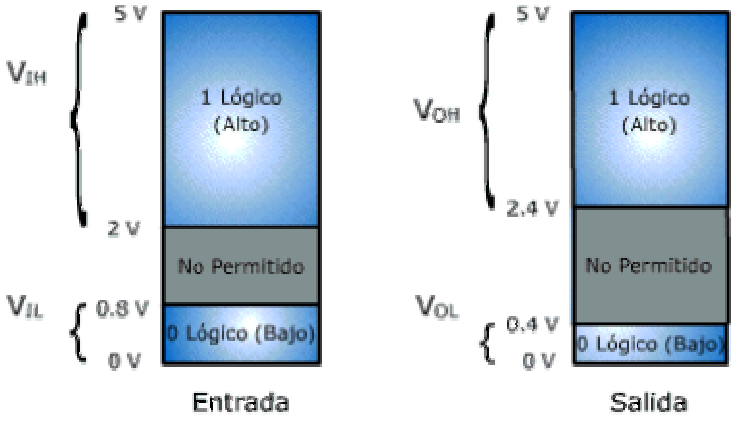
V_{IH} (min) Voltaje de entrada de alto nivel: valor de tensión mínimo admitido que se requiere en la entrada para un “1” lógico.

V_{IL} (max) Voltaje de entrada de bajo nivel: valor de tensión máximo admitido que se requiere en la entrada para un “0” lógico.

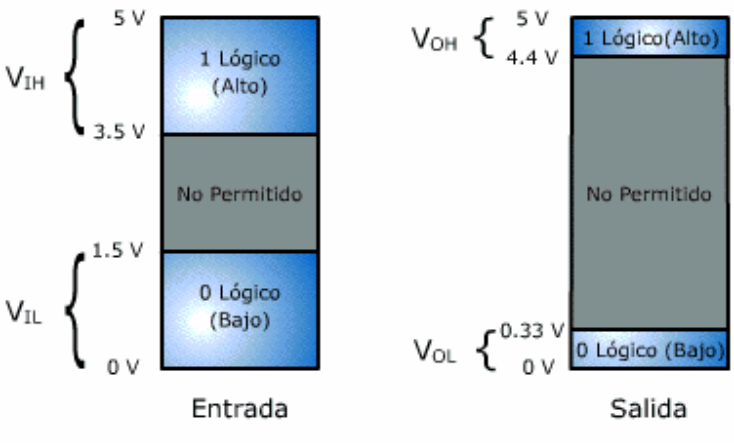
V_{OH} (min) Voltaje de salida de alto nivel: valor de tensión mínimo en la salida para un estado de “1” lógico.

V_{OL} (max) Voltaje de salida de bajo nivel: valor de tensión máximo en la salida para un estado de “0” lógico.

Niveles Lógicos TTL



Niveles Lógicos CMOS



V_{NH} Margen de ruido en el estado alto:

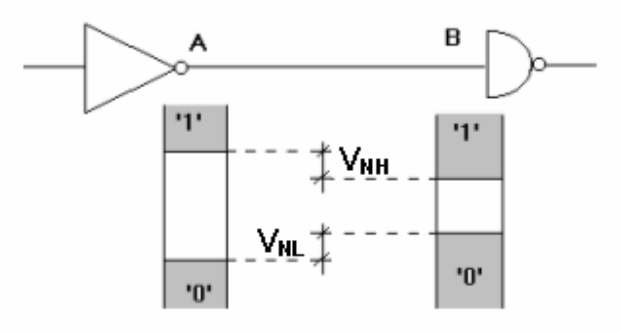
$$V_{NH} = V_{OH}\ (min) - V_{IH}\ (min)$$

El valor **V_{OH}** en **A** podrá disminuir **V_{NH}** voltios antes de que la entrada **B** deje de reconocer un "1" lógico.

V_{NL} Margen de ruido en el estado bajo:

$$V_{NL} = V_{IL}\ (max) - V_{OL}\ (max)$$

El valor **V_{OL}** en **A** podrá aumentar **V_{NL}** voltios antes de que la entrada **B** deje de reconocer un "0" lógico.



I_{IH} Corriente de entrada de alto nivel: corriente que fluye en una entrada cuando se le aplica una tensión V_{IH}.

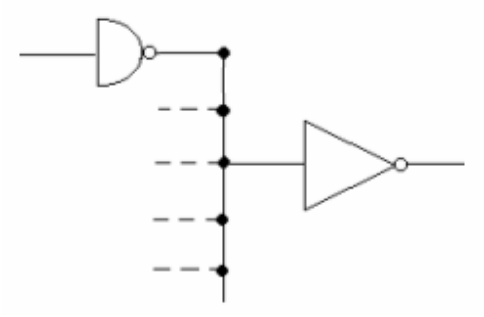
I_{IL} Corriente de entrada de bajo nivel: corriente que fluye en una entrada cuando se le aplica una tensión V_{IL}.

I_{OH} Corriente de salida de alto nivel: corriente que fluye desde la salida en el estado "1" lógico.

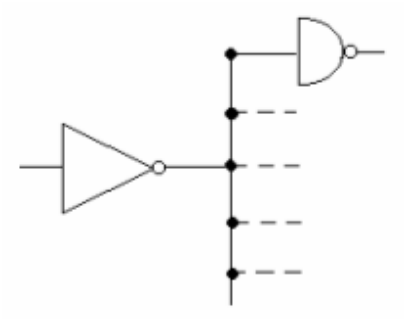
I_{OL} Corriente de salida de bajo nivel: corriente que fluye desde la salida en el estado "0" lógico.

I_{cc} Corriente que se toma de la fuente de alimentación: varía de acuerdo a los estados lógicos del circuito.

FAN-IN Cargabilidad de la entrada: numero máximo de salidas lógicas que se pueden conectar en una entrada.

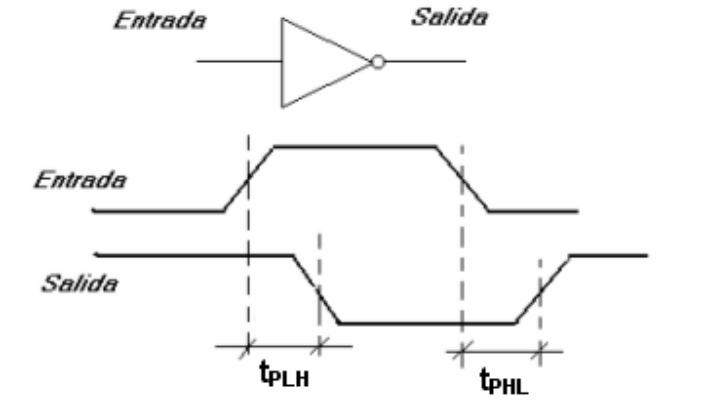


FAN-OUT Cargabilidad de la salida: numero máximo de entradas lógicas que se pueden conectar en una salida.



t_{PLH} Tiempo de retardo para pasar de un estado lógico “0” a “1”.

t_{PHL} Tiempo de retardo para pasar de un estado lógico “1” a “0”.



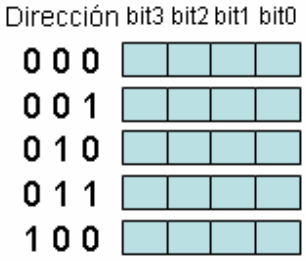
PARAMETRO		FAMILIA LOGICA	
DESCRIPCION	**UNIDAD**	**TTL estándar**	**CMOS estándar**
Fan -Out	unidades	10	50
Retardo	ns	10	40
Desempeño al ruido	V	0,4	1
Disipación de potencia	µw	10.000	0,01
Tensión de alimentación	v	$5 \pm 0{,}25$	$5 \pm 0{,}25$

Unidad De Memoria

Dispositivo de almacenamiento de información.
Formadas por un conjunto de registros con una dirección asociada.

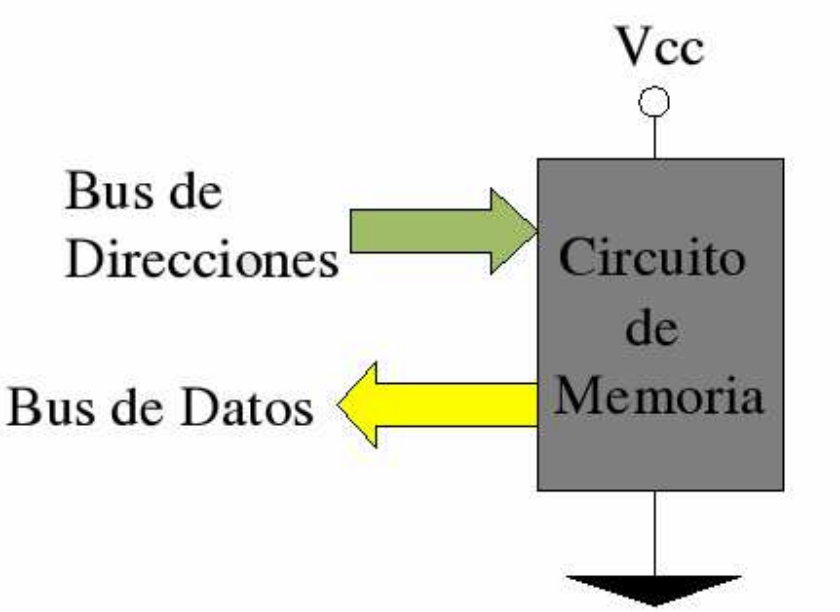
Ejemplo

Memoria formada por 5 registros de 4 bits cada uno.

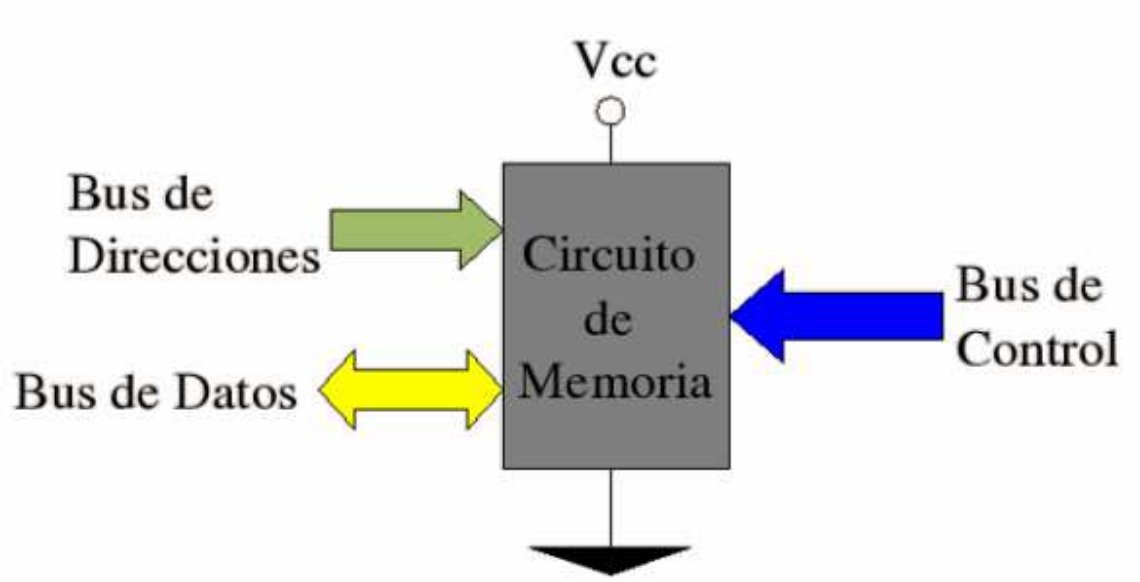


Capacidad de almacenamiento = 20 bits

Circuito general de memoria de solo lectura



Circuito general de memoria de lectura y escritura



Clasificaciones

Según la permanencia de la información

Volátil

Necesita estar alimentada eléctricamente para almacenar la información.

- Estática:

La información permanece almacenada por tiempo indefinido mientras estén alimentadas eléctricamente. No se necesita rescribirla periódicamente.

- Dinámica:

La información no permanece almacenada por tiempo indefinido mientras estén alimentadas eléctricamente. Se necesita rescribirla periódicamente.

> El tiempo de acceso a la memoria dinámica es mayor que a la estática. Pero el tamaño y costo de la memoria dinámica es menor que el de la estática.

No volátil

No necesita estar alimentada eléctricamente para almacenar la información.

Según la tecnología

Magnética

- Disco flexible.
- Disco duro.
- Cinta magnética.

Óptica

- CD.
- DVD.
- Blu-ray.

Semiconductora

Construidas a partir de semiconductores de silicio.

Según el tipo de acceso

Secuencial

Funcionan como la grabación y reproducción de un cassette. Si deseo alcanzar una determinada grabación, debo pasar por todas las grabaciones anteriores.

Aleatorio

Van directamente a buscar los datos a la dirección correspondiente.

Según la función o el tiempo de lectura y escritura

- Solo lectura (ROM)

te>>tl

- Solo lectura y programable (PROM)
- Solo lectura, programable y borrable con luz ultravioleta (EPROM)
- Solo lectura, programable y borrable eléctricamente (EEPROM)
- Solo lectura, programable, borrable eléctricamente y rápida (FLASH)

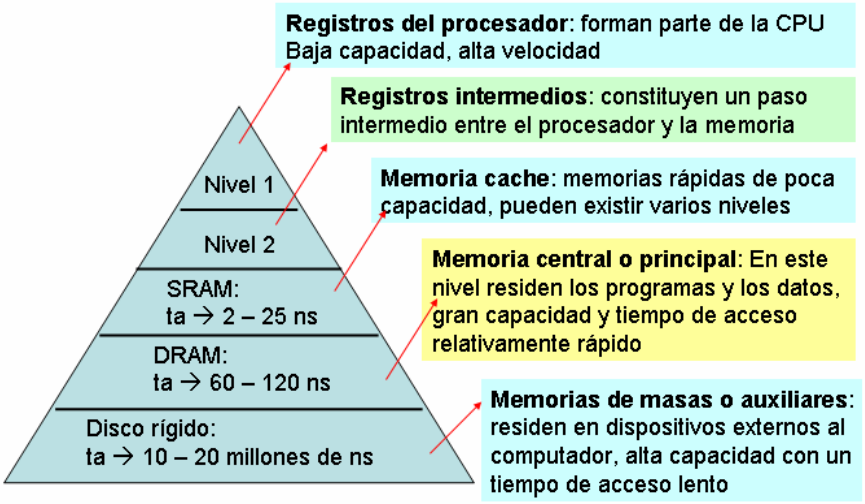
te=tl

- Lectura y escritura (RWM)

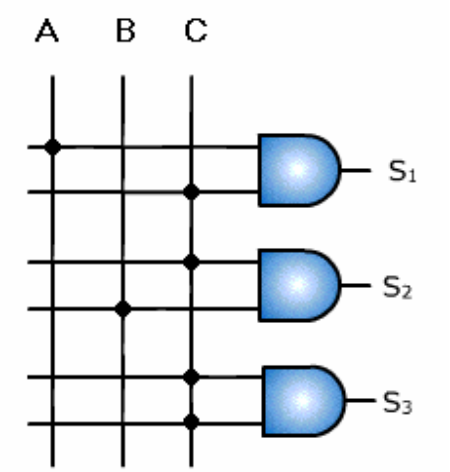
Según el tipo de información que almacenen

- De Datos
- De Programa

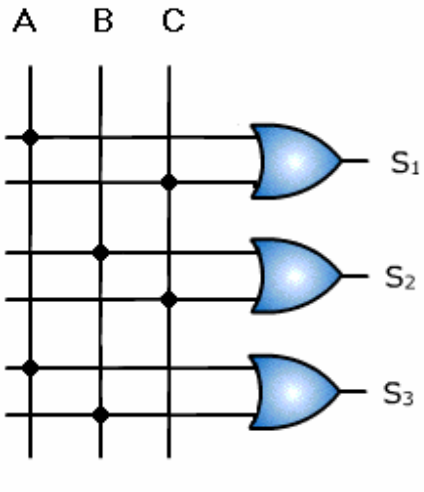
Según la jerarquía



Circuito de memoria de solo lectura



DIRECCIÓN			CONTENIDO		
A	B	C	S1	S2	S3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	1



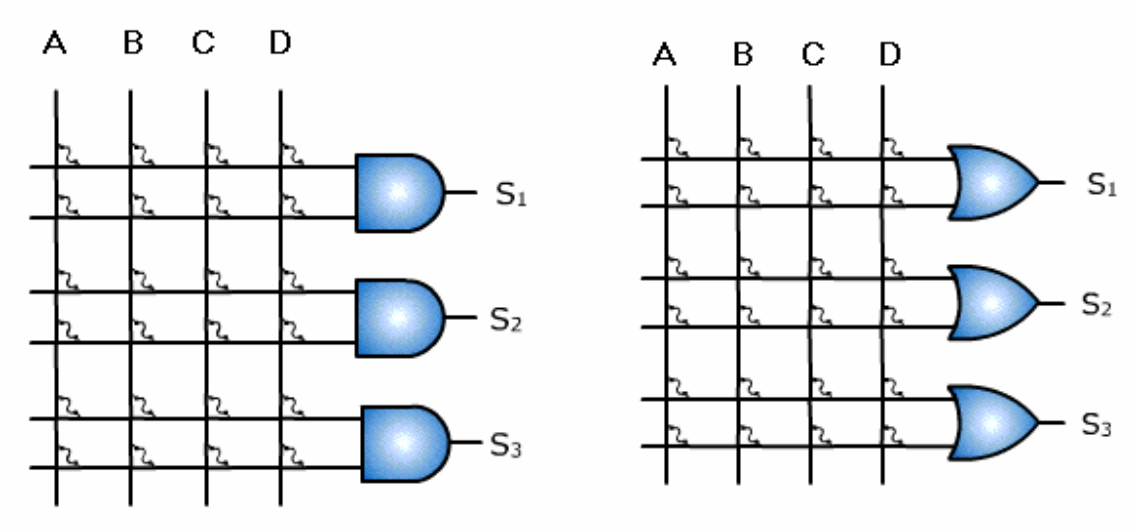
DIRECCIÓN			CONTENIDO		
A	B	C	S1	S2	S3
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Memorias formadas por 8 registros de 3 bits cada uno.

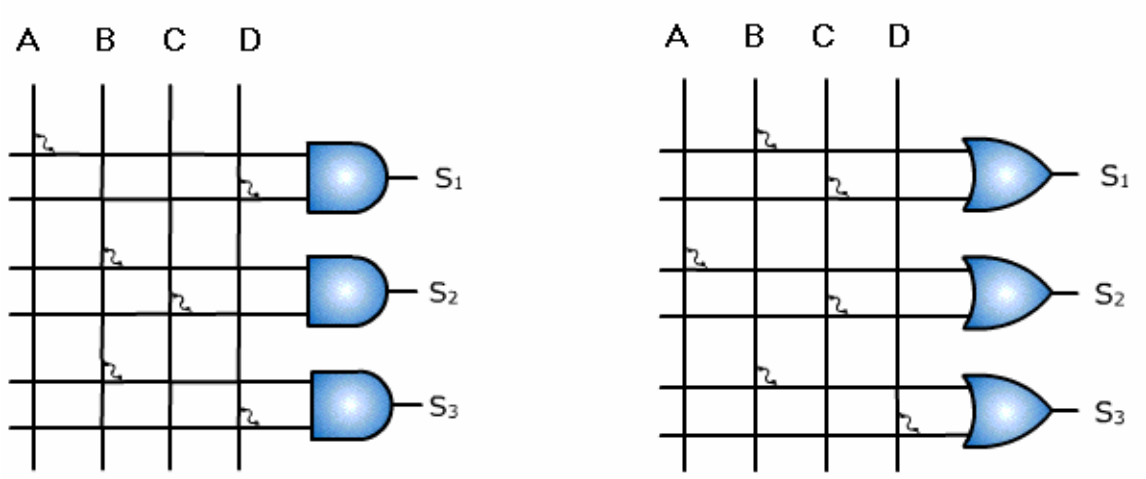
Capacidad de almacenamiento = 24 bits

Circuito de memoria de solo lectura programable

No programadas



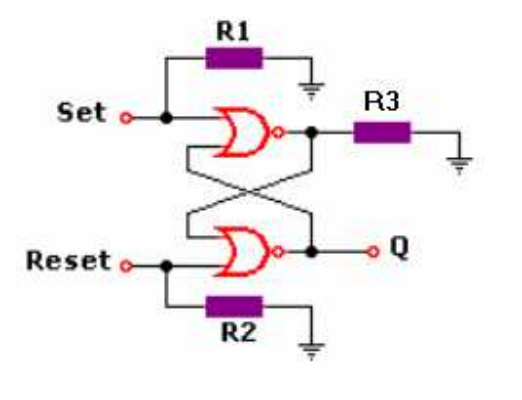
Programadas



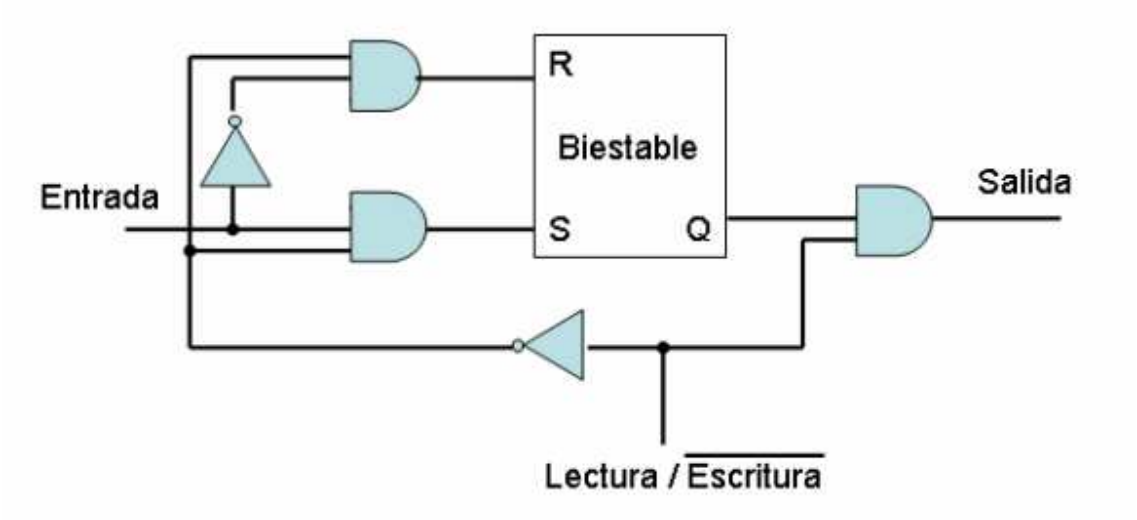

Fusible

Circuito de Memoria de lectura y escritura

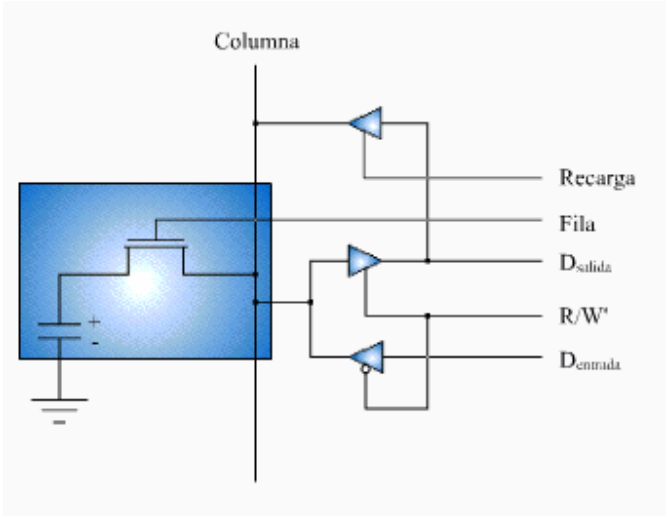
Estática

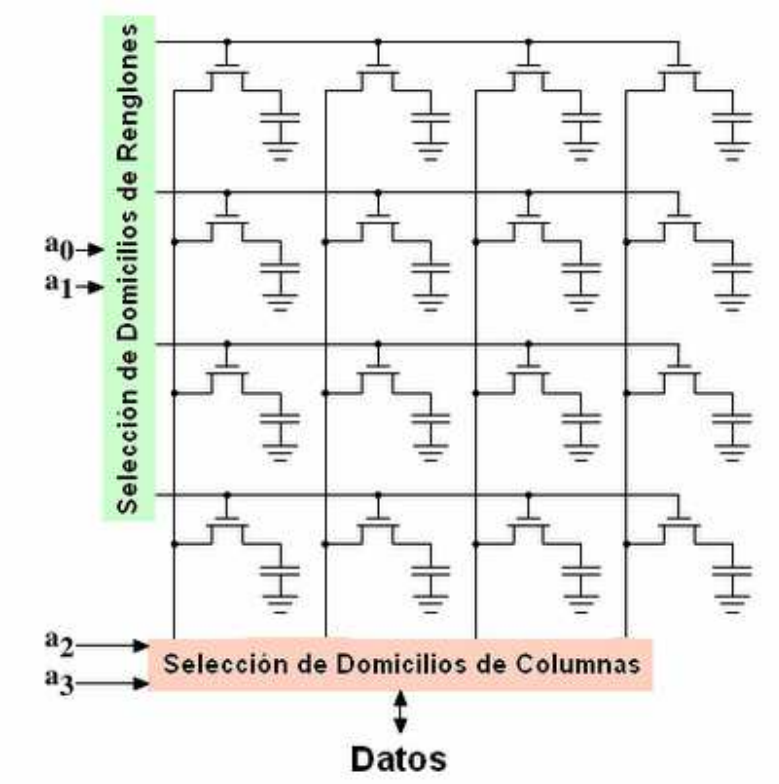


Entrada actual		Salida actual	Salida siguiente
Set	Reset	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	---
1	1	1	---



Dinámica

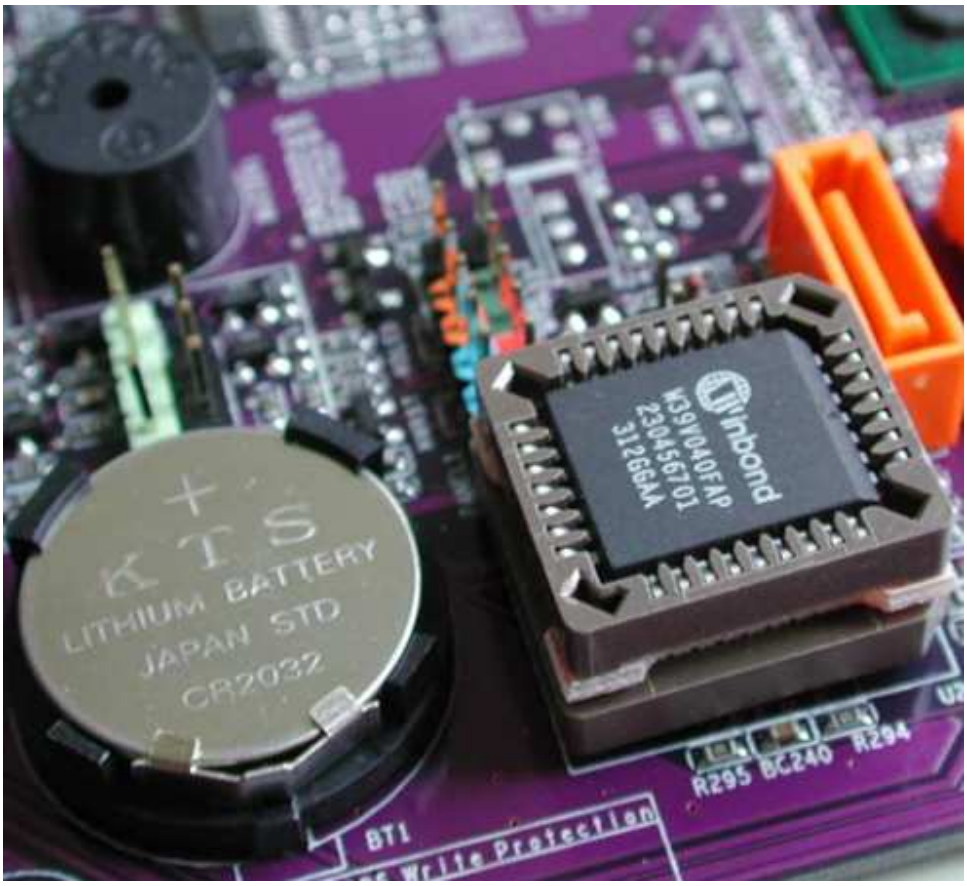

Columna
Recarga
Fila
D_{salida}
R/W'
$D_{entrada}$


Selección de Domicilios de Renglones
a_0
a_1
a_2
a_3
Selección de Domicilios de Columnas
Datos

Unidad De Entrada Y Salida

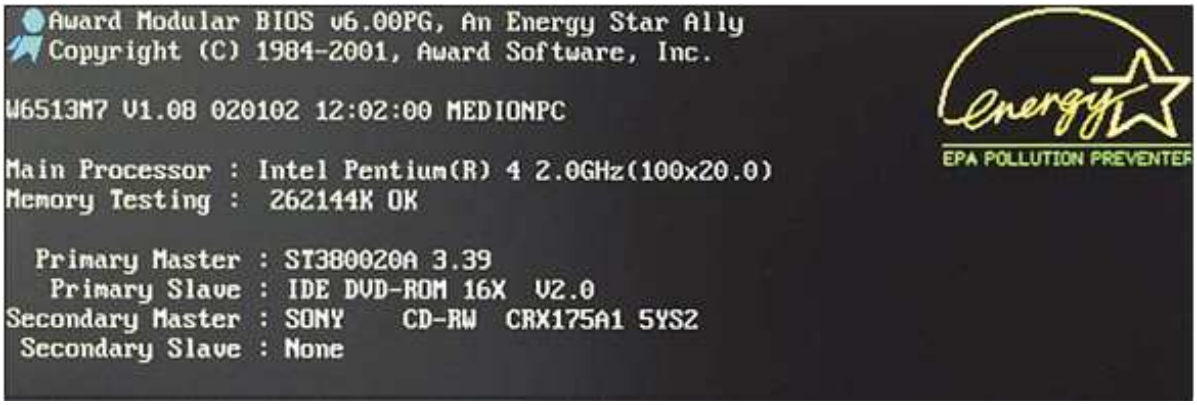
Sistema Básico De Entrada Y Salida (BIOS)

Programa interfaz entre el software y el hardware de la computadora. Almacenado en una memoria de solo lectura. Su configuración se almacena en una memoria volátil de lectura y escritura alimentada por una pila.



POST (Autocomprobación De Arranque)

Programa que comprueba automáticamente al encender la computadora los distintos componentes de hardware (microprocesador, placa de video, memoria, disco rígido, teclado) y realiza un conteo de la memoria RAM. Si encuentra algún problema lo indica con algún mensaje por pantalla o a trabes de un código sonoro.

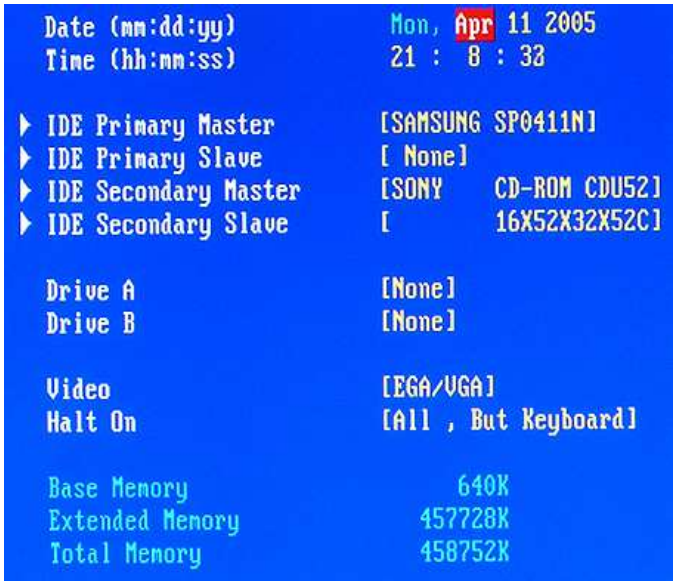


SETUP (Organización)

Programa operado por un menú grafico que se activa al oprimir una determinada tecla (generalmente **supr** o **F2**) durante la ejecución del POST que permite determinar la configuración de la computadora: el motherboard, el reloj interno, unidades de disco y secuencia de unidades de arranque, contraseñas, etc.

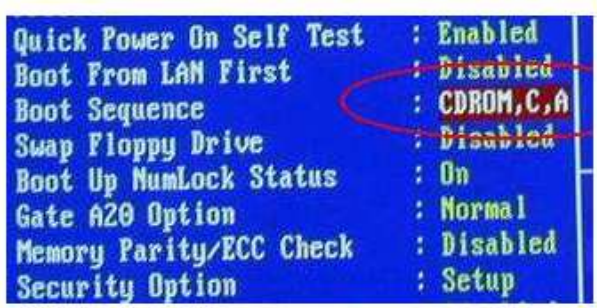
Configuración Principal

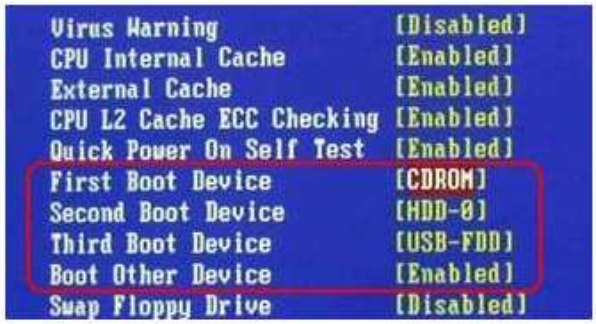
- Standard Features (Características estándar)



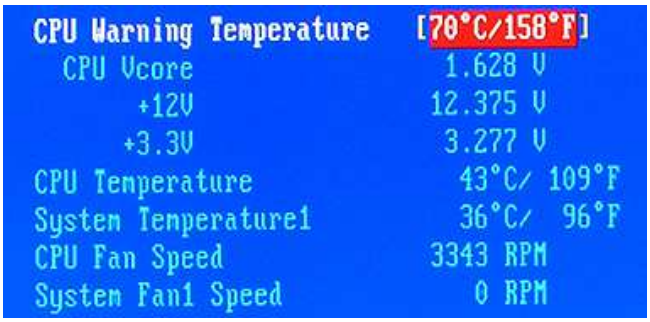
- Boot Sequence (Secuencia de arranque)

Indica al BIOS a qué unidad ir a buscar el arranque del sistema operativo.

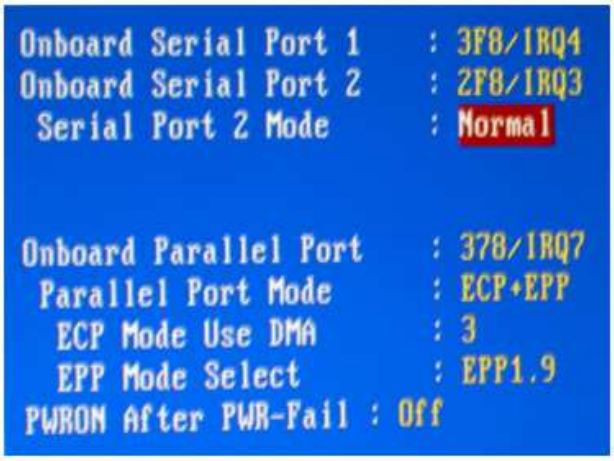


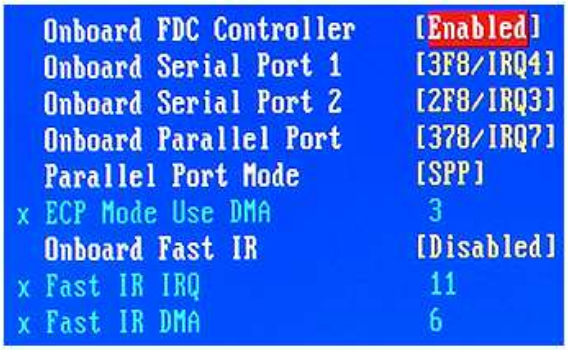


- PC Health Status (Estado de salud de la Pc)

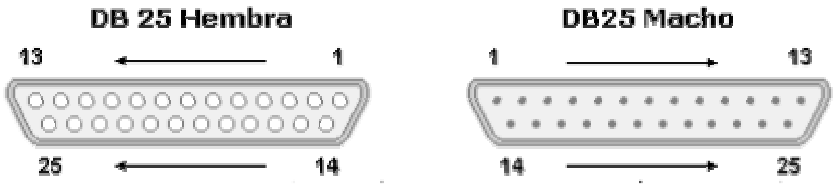


- Ports (Puertos)

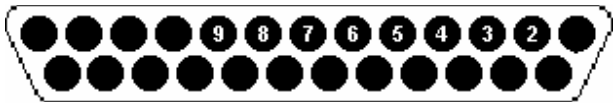




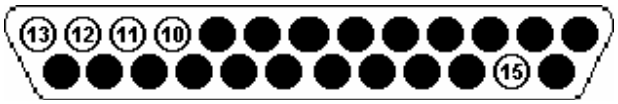
Puerto Paralelo



Escritura De Datos								
Dirección LPT1 0x378 Dirección LPT2 0x278								
DATO	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DB25	Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2



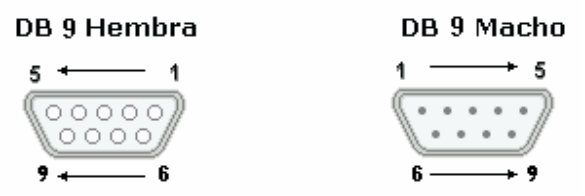
Lectura De Datos								
Dirección LPT1 0x379 Dirección LPT2 0x279								
DATO	~ BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DB 25	Pin 11	Pin 10	Pin 12	Pin 13	Pin 15	No usar	No usar	No usar



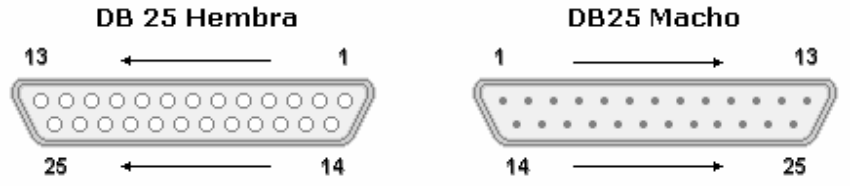
Características Eléctricas

- Tensión de nivel alto
  3,8 a 5 V
- Tensión de nivel bajo
  0 a 0,8 V
- Corriente de salida máxima
  2,6 mA
- Corriente de entrada máxima
  24 mA

Puerto Serie



PIN	NOMBRE	
1	CD	CARRIER DETECT
2	RD	RECEIVE DATA
3	TD	TRANSMIT DATA
4	DTR	DATA TERMINAL READY
5	SG	SIGNAL GROUND
6	DSR	DATA SET READY
7	RTS	REQUEST TO SEND
8	CTS	CLEAR TO SEND
9	RI	RING INDICATOR



PIN	NOMBRE	
1	GND	Tierra del sistema, blindaje
2	TD	TRANSMIT DATA
3	RD	RECEIVE DATA
4	RTS	REQUEST TO SEND
5	CTS	CLEAR TO SEND
6	DSR	DATA SET READY
7	SG	SIGNAL GROUND
8	CD	CARRIER DETECT
9		Reservado
10		Reservado
11		No usado
12	SCD	Detección de portadora secundario
13	SCTS	Solicitud de envío secundaria
14	STD	Transmisión de datos secundaria
15	TxD	Transmisión de datos
16	SRD	Recepción de datos secundario
17		Reloj de recepción
18		No usado
19	SRTS	Solicitud de envío secundaria
20	DTR	DATA TERMINAL READY
21		Detección de calidad de señal
22	RI	RING INDICATOR
23	DRS	Selección de tasa de datos
24		Reloj de transmisión
25		No usado

Tierra de referencia (SG = Signal Ground): es la línea de referencia de tensión de señal.

Salida de datos (TD o TxD): Por esta línea salen los datos de la computadora emisora. Para que esto sea posible es necesario que estén activas las líneas RTS, DTR, CTS Y DSR.

Recepción de datos (RD o RxD): Entrada de datos serie a la computadora enviados por el módem.

Indicador de llamada (RI): Por esta línea el módem le indica a la computadora que ha recibido una señal de llamada telefónica.

Detección de portadora (DCD): A través de esta línea, el módem informa a la computadora que se ha establecido una comunicación telefónica con otro módem.
De hecho, el módem de recepción activa DCD si ha detectado una señal de portadora proveniente del terminal remoto, y entonces se establecerá una conexión.
DCD permanece activa mientras la conexión siga establecida. Si la comunicación es unidireccional, solo el módem de recepción activa DCD.

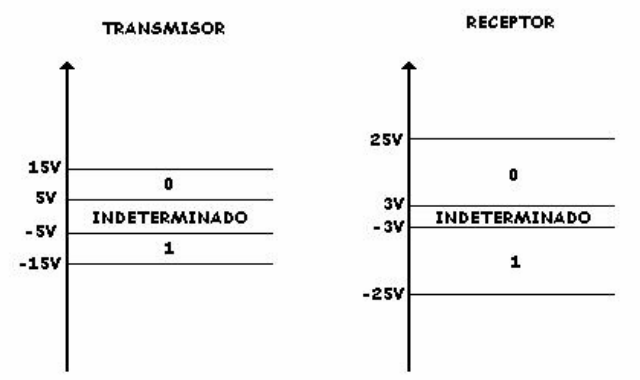
Módem listo (DSR): El módem informa activando esta línea, que está encendido, que ha completado el proceso de inicialización para establecer una conexión remota y puede comunicarse con la computadora.

Terminal listo (DTR): Por esta línea la interfaz RS232 informa al módem que está encendida y preparada.
DTR debe estar activa durante toda la comunicación. Las líneas DTR y DSR son responsables de mantener la comunicación. Si la línea DTR no está activada, el módem no enviará ni aceptará datos.

Solicitud de envío (RTS): Mediante esta línea se informa al módem que la computadora intenta enviarle un dato.

Libre para recibir (CTS): El módem responde al pedido de RTS de la computadora, activando la línea CTS a fin de permitir el envío solicitado del dato.
Las líneas RTS y CTS son las responsables de la transferencia de datos de transmisión.

Características Eléctricas



Norma RS232

Estandarización de la transmisión serie asincrónica

Estado de reposo: El estado ocioso de la línea corresponde al estado del "1".

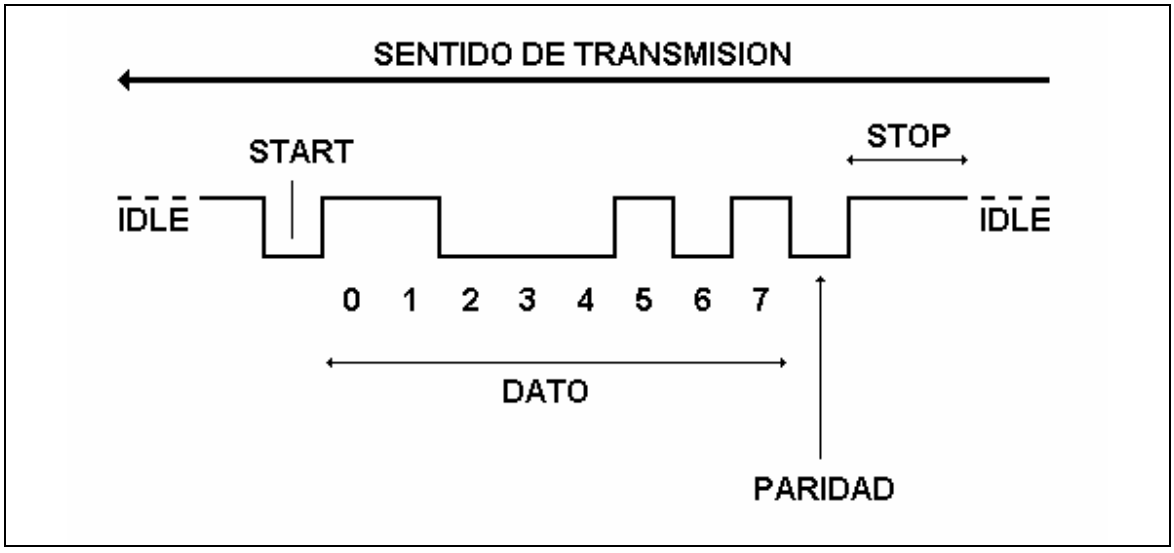
Bit de arranque: Tiene que tener un estado opuesto al anterior, y por lo tanto se corresponde con un estado "0".

Cantidad de bits de información: Se puede optar por utilizar 7 u 8 bits de la misma duración. La secuencia de transmisión comienza con el bit menos significativo.

Bit de paridad: Constituye un método simple de detección de errores. Consiste en colocar este bit en "0" o en "1" de manera tal que la cantidad de bits en "1" considerando los de información y el de paridad sea, o bien par, o bien impar. Si algún bit cambiara erróneamente durante la transmisión, resultará que la paridad en el extremo receptor no coincidirá con la acordada, detectándose un error.
La presencia de este bit es opcional, como así también la paridad acordada.

Bits de Stop: Se pueden seleccionar 1, 1 ½ o 2 bits de stop. Estos bits tienen el estado "1" y restituyen la línea al estado de reposo. Su función es la de enmarcar la secuencia de bits de información.

Estructura de un caracter



En la figura se muestra el envío de un carácter con la siguiente convención:

Cantidad de Bits del Dato: 8
Dato transmitido: 10100011
Cantidad de bits de stop: 2
Bit de paridad: SI
Tipo de paridad: PAR

UART

Dispositivo especializado para manejar las comunicaciones serie asincrónicas.

La UART realiza la conversión de datos de formato serie a paralelo en recepción, y de paralelo a serie en transmisión.
La CPU puede leer el status de funcionamiento de la UART en cualquier momento, dado que esta dispone de un registro para tal fin.
La información de status incluye el tipo y en que se encuentran las transmisiones que se están llevando a cabo, la existencia o no de errores y situaciones de *break.*
La UART necesita la provisión de un reloj externo de referencia. Realiza una división interna por 16 de la frecuencia de este clock, lo que le permite minimizar errores de fase, y detección de un falso bit de arranque, como se verá posteriormente.
Esta nueva frecuencia es posible dividirla mediante programación del integrado, por valores comprendidos entre 1 y 65535, de manera que la tasa de generación de baudios sea seleccionable por el usuario.
La UART tiene completa capacidad de control sobre el módem y las interrupciones al procesador relacionadas. Estas pueden ser programadas según los requerimientos de la aplicación a fin de minimizar los requerimientos de procesamiento por parte de la CPU en cuanto a la comunicación serie.
Es posible seleccionar por programación la longitud del dato (5, 6, 7, u 8 bits), la paridad (par, impar o ninguna) y la cantidad de bits de stop (1,1 ½ o 2).

Puertos de comunicación serie

La PC presenta cuatro puertos standard de comunicación serie denominados COM1, COM2, COM3 y COM4.
Como vimos anteriormente, la UART se muestra como un conjunto de 8 ports de los cuales, el primero estará ubicado en la posición base asociada a COMi y los demás en orden de offset creciente.
Asimismo cada uno de los puertos tiene asociada una interrupción a la CPU en el mapa de las IRQ (Interrupt Request).
A continuación se indican cuales son las direcciones standard de los puertos serie y las interrupciones asociadas:

NOMBRE	Dirección Base	IRQ
COM1	3F8	4
COM2	2F8	3
COM3	3E8	4
COM4	2E8	3

En el área de datos correspondiente al BIOS se pueden encontrar las direcciones de ports asignadas a cada uno de los puertos de comunicación serie.
Las direcciones de esta tabla son las siguientes:

DIRECCIÓN	CONTENIDO
0000:0400	Dirección Base de COM1
0000:0402	Dirección Base de COM2
0000:0404	Dirección Base de COM3
0000:0406	Dirección Base de COM4

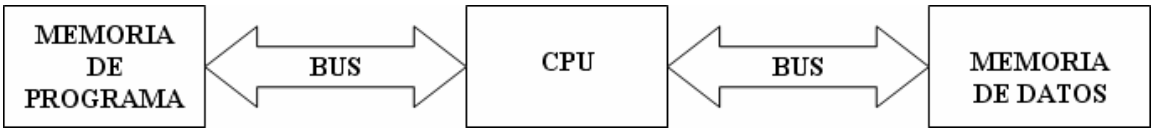
Arquitecturas De Computadoras

Forma de seleccionar e interconectar componentes de hardware para crear computadoras según ciertos requerimientos.

Harvard

- El término proviene de la computadora Harvard Mark I.
- La primera computadora con esta arquitectura fue la Harvard Mark I.
- Se utiliza en supercomputadoras, microcontroladores, y sistemas integrados.

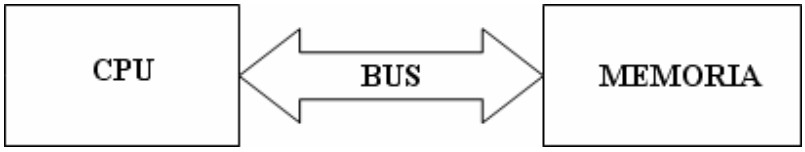
- Existe una memoria donde se almacenan las instrucciones de programa y una memoria donde se almacenan los datos.
- Existe un bus de direcciones, de datos y de control por cada memoria.
- Cuando la CPU se dirige a la memoria principal, extrae la instrucción y simultáneamente los datos con los que se efectúa la instrucción.
- La longitud de los datos e instrucciones no necesariamente debe ser la misma.

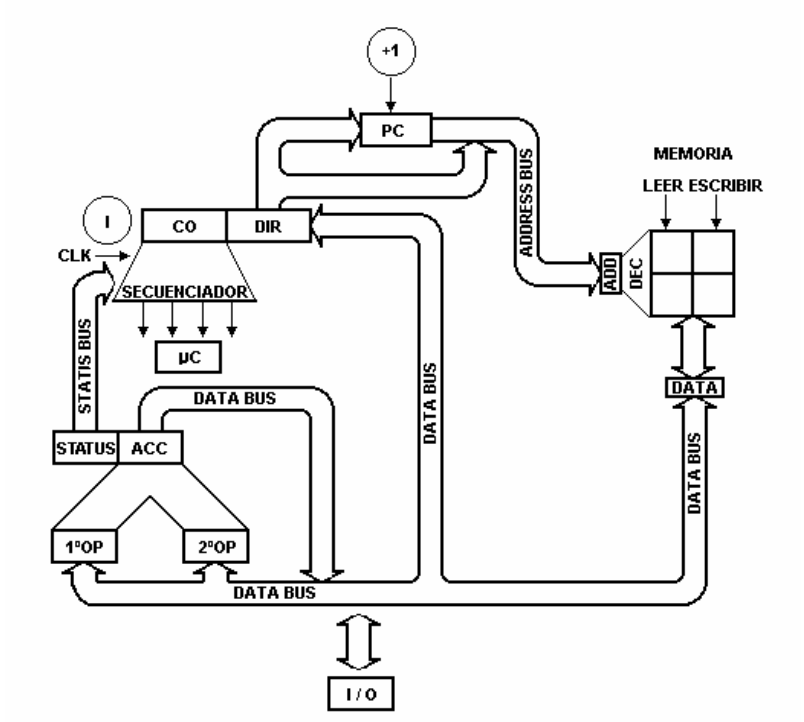


Von Neumann

- El término proviene del apellido de un matemático.
- La primera computadora con esta arquitectura fue la ENIAC (Computador e Integrador Numérico Electrónico).
- Se utiliza en las computadoras personales.

- Existe una única memoria donde se almacenan conjuntamente las instrucciones de programa y los datos.
- Existe un unico bus de direcciones, de datos y de control.
- Cuando la CPU se dirige a la memoria principal, extrae la instrucción y después los datos con los que se efectúa la instrucción.
- La longitud de los datos e instrucciones debe ser la misma.

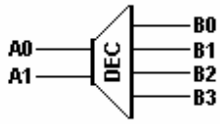




Unidad De Memoria

Dispositivo que almacena información (Datos e Instrucciones).

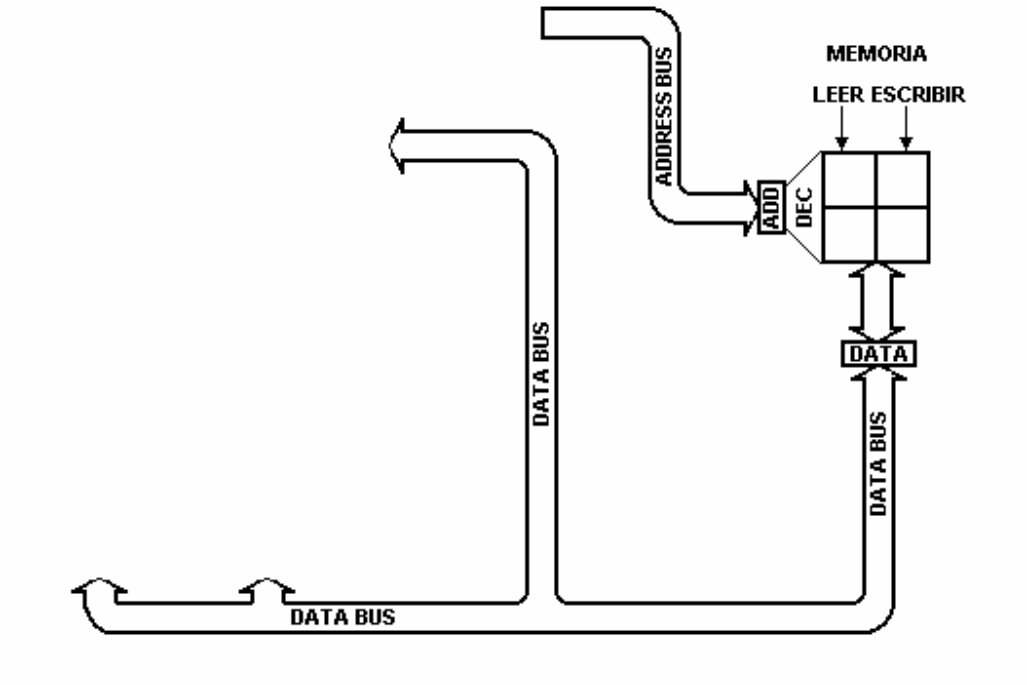
Decodificador (DEC) → Cada salida habilita, según la combinación de bits en su entrada, la posición de memoria correspondiente para intercambiar (leer o escribir) información, según indiquen los microcomandos. Tiene tantas salidas como combinaciones de bits posibles en su entrada.



ENTRADAS		SALIDAS			
A0	A1	B0	B1	B2	B3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Registro de direcciones (ADD) → Almacena la dirección de memoria donde se intercambiará información.
Bus de direcciones (ADDRES BUS)→ Comunica el campo de dirección con el registro de direcciones.

Registro de datos (DATA) → Almacena la información que se intercambiara desde o hacia la memoria.
Bus de datos (DATA BUS) → Envía la instrucción a ejecutar desde el registro de datos al registro de instrucción. Envía los datos a tratar desde el registro de datos a los registros de entrada de la ALU. Envía los resultados del registro de salida de la ALU al registro de datos y/o a los registros de entrada de la ALU.



Unidad Central De Procesamiento

- **Unidad de Control (UC)** → Emite microcomandos **(μC)** :

Controla las transferencias de los datos y las instrucciones desde y hacia la CPU.

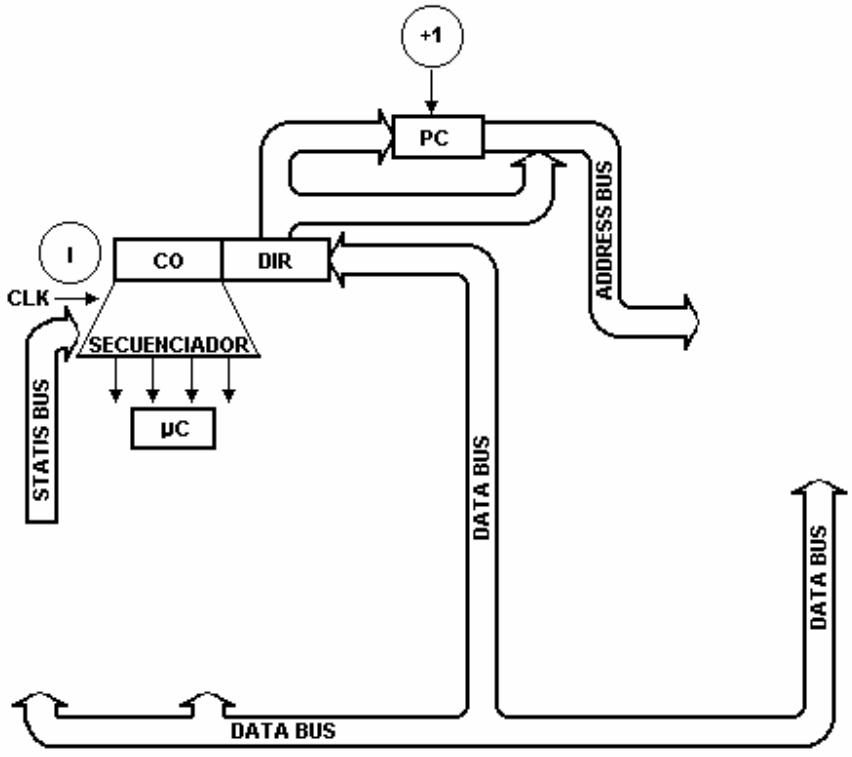
Registro de instrucción (I) → Almacena la instrucción a ejecutar.

Campos De La Instrucción
Código de operación (CO) → Representa el tipo de operación a realizar.
Campo de dirección (DIR) → Representa los operandos o sus direcciones.

Tipos De Instrucciones	**Descripción**
Transferencia	Los datos pueden transferirse de los registros de la CPU a la memoria o las unidades de E/S o viceversa.
Procesamiento	La CPU realiza alguna operación aritmética o lógica sobre los operandos.
Control	Una instrucción puede alterar el orden de ejecución de un programa modificando el valor del PC.

Secuenciador y el reloj de la maquina (CLK) → Determinan el orden temporal y la secuencia de los microcomandos.

Registro contador de programa (PC) → Almacena la dirección de memoria de la próxima instrucción a ejecutar y le llega el microcomando "+1" que incrementa en uno su contenido.

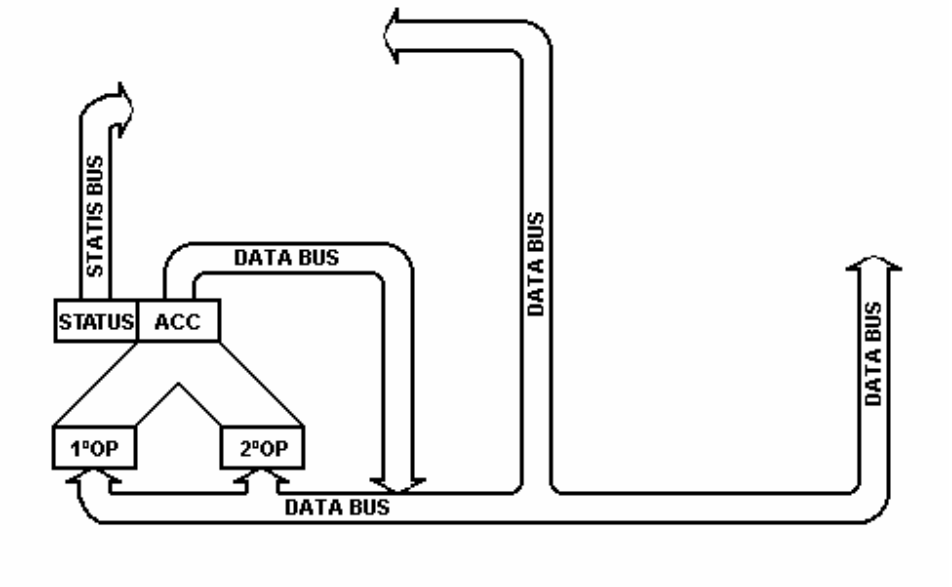


- **Unidad lógica aritmética (ALU)** → Realiza operaciones aritméticas (suma, resta, multiplicación, división, …) y lógicas (and, or, not, …).

Registros de entrada: 1° operando (1°OP) y 2° operando (2°OP) → Almacenan los datos a tratar por la ALU.

Registro de salida: acumulador (ACC) → Almacena el resultado de la ALU.

Registro de estados (STATUS) → Almacena características del resultado de la ALU (si es o no cero, si es positivo o negativo, …) útiles para tomar decisiones.
Bus de estados (STATUS BUS) → Comunica la unidad de control con el registro de estados.



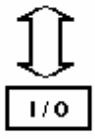
Unidad De Entrada Y Salida

Permiten que la maquina interactúe con el mundo exterior a través de los periféricos:

- De solo entrada (teclado, mouse, micrófono, …)
- De solo salida (monitor, impresora, parlante, …)
- De entrada y salida (disquete, cd grabable, disco rígido, …)

Modulo De Entrada Y Salida

Registro de direcciones de entrada y salida (I/O) → Almacena la dirección de memoria donde se intercambiara información.
Registro de datos de entrada y salida (I/O) → Almacena la información que se intercambiara desde o hacia el periférico.



Ejemplo 1

Sumar 2 números y el resultado guardarlo en memoria

Fase de búsqueda de la 1º instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Cargar el primer operando con el primer dato	ADD ← DIR LEER 1ºOP ← DATA
Fase de búsqueda de la 2º instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Cargar el segundo operando con el segundo dato	ADD ← DIR LEER 2ºOP ← DATA
Fase de búsqueda de la 3º instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Sumar (coloca en el acumulador la suma de los dos operandos)	ACC←1ºOP+2ºOP
Fase de búsqueda de la 4º instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Guardar el resultado en una dirección de memoria	ADD ← DIR ESCRIBIR DATA ← ACC

Ejemplo 2

Buscar un valor en memoria, sumarlo al contenido actual del acumulador y el resultado enviarlo al registro de entrada/salida.

Fase de búsqueda de la 1° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Cargar el primer operando con el primer dato	ADD ← DIR LEER 1°OP ← DATA
Fase de búsqueda de la 2° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Cargar el segundo operando con el segundo dato	2°OP ← ACC
Fase de búsqueda de la 3° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Sumar (coloca en el acumulador la suma de los dos operandos)	ACC←1°OP+2°OP
Fase de búsqueda de la 4° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Enviar el resultado al registro entrada/salida	I/O ← ACC

Ejemplo 3

Buscar un valor en memoria, ingresar otro desde el registro de entrada/salida, compararlos y si son iguales saltar si no continuar con el orden original.

Fase de búsqueda de la 1° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Cargar el primer operando con el primer dato	ADD ← DIR LEER 1°OP ← DATA
Fase de búsqueda de la 2° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Cargar el segundo operando con el segundo dato	2°OP ← I/O
Fase de búsqueda de la 3° instrucción	ADD ← PC LEER I ← DATA
Preparación para la siguiente instrucción	PC ← PC + 1
Comparar	si son iguales PC ← DIR si son distintos PC ← PC

UNIDAD Nº2

Sistemas Posicionales De Numeración

Una cantidad se representa por una cadena de elementos:

$$N_b = d_n ... d_3 d_2 d_1$$

Cada uno de estos elementos tiene un valor asociado a la posición que ocupa dentro de la cadena.

$$N_b = \sum_{i=1}^{n} d_i * b^{i-1} = d_1 * b^0 + ... + d_n * b^{n-1}$$

- d_i → Digito cualquiera de los permitidos por el sistema de numeración.
- n → Cantidad de dígitos del número.
- b → Base del sistema de numeración.
- S → Conjunto de símbolos permitidos por el sistema.

Sistema decimal

$b = 10$

$S = \{0,1,2,3,4,5,6,7,8,9\}$

Sistema binario

$b = 2$

$S = \{0,1\}$

Sistema octal

$b = 8$

$S = \{0,1,2,3,4,5,6,7\}$

Sistema hexadecimal

$b = 16$

$S = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

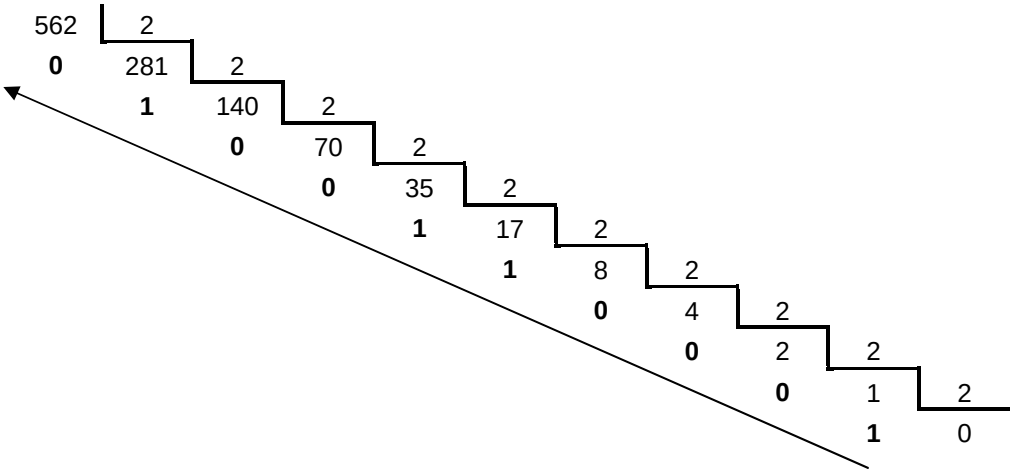
$C = b^n$ → Cantidad de elementos representables

Conversiones Entre Sistemas

Decimal → Binario

- Divisiones sucesivas.

562_{10}



$562_{10} = 1000110010_2$

Binario → Decimal

- Sumatoria.

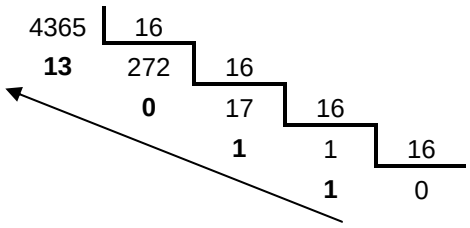
$101101_2 = 1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0$

$101101_2 = 45_{10}$

Decimal → Hexadecimal

- Divisiones sucesivas.

4365_{10}



$4365_{10} = 110D_{16}$

Hexadecimal → Decimal

- Sumatoria.

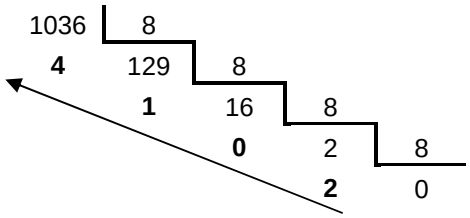
$7C1_{16} = 7 \times 16^2 + 12 \times 16^1 + 1 \times 16^0$

$7C1_{16} = 1985_{10}$

Decimal → Octal

- Divisiones sucesivas.

1036_{10}



$1036_{10} = 2014_8$

Octal → Decimal

- Sumatoria.

$3701_8 = 3 \times 8^3 + 7 \times 8^2 + 0 \times 8^1 + 1 \times 8^0$

$3701_8 = 1985_{10}$

Binario → Hexadecimal

- Agrupando de a 4.

$$101010011111111_2 = \underbrace{0010}_{2}\underbrace{1010}_{A}\underbrace{0111}_{7}\underbrace{1111}_{15}{}_2$$

$$101010011111111_2 = 2A7F_{16}$$

Hexadecimal → Binario

- Representando de a 4.

$$31D_{16} = \underbrace{0011}_{3}\underbrace{0001}_{1}\underbrace{1101}_{D}{}_2$$

Binario → Octal

- Agrupando de a 3.

$$101010011111111_2 = \underbrace{010}_{2}\underbrace{101}_{5}\underbrace{001}_{1}\underbrace{111}_{7}\underbrace{111}_{7}{}_2$$

$$101010011111111_2 = 25177_8$$

Octal → Binario

- Representando de a 3.

$$563_8 = \underbrace{101}_{5}\underbrace{110}_{6}\underbrace{011}_{3}{}_2$$

Decimal				Binario				Octal				Hexadecimal			
10^3	10^2	10^1	10^0	2^3	2^2	2^1	2^0	8^3	8^2	8^1	8^0	16^3	16^2	16^1	16^0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	2	0	0	1	0	0	0	0	2	0	0	0	2
0	0	0	3	0	0	1	1	0	0	0	3	0	0	0	3
0	0	0	4	0	1	0	0	0	0	0	4	0	0	0	4
0	0	0	5	0	1	0	1	0	0	0	5	0	0	0	5
0	0	0	6	0	1	1	0	0	0	0	6	0	0	0	6
0	0	0	7	0	1	1	1	0	0	0	7	0	0	0	7
0	0	0	8	1	0	0	0	0	0	1	0	0	0	0	8
0	0	0	9	1	0	0	1	0	0	1	1	0	0	0	9
0	0	1	0	1	0	1	0	0	0	1	2	0	0	0	A
0	0	1	1	1	0	1	1	0	0	1	3	0	0	0	B
0	0	1	2	1	1	0	0	0	0	1	4	0	0	0	C
0	0	1	3	1	1	0	1	0	0	1	5	0	0	0	D
0	0	1	4	1	1	1	0	0	0	1	6	0	0	0	E
0	0	1	5	1	1	1	1	0	0	1	7	0	0	0	F

Codificación

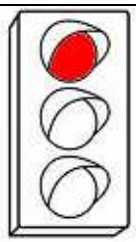
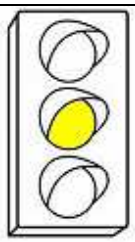
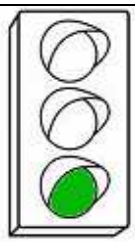
Es la representación de ciertos elementos, a través de la asignación a cada uno de ellos mediante una combinación determinada de símbolos (llamada palabra del código), elegidos de un juego permitido de símbolos (llamado alfabeto del código). En los códigos binarios el alfabeto del código es sobre los símbolos 0 y 1.

n = cantidad de dígitos de la palabra del código

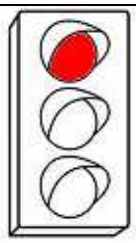
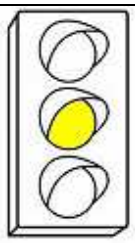
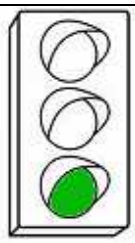
Ejemplo

Consideremos el estado de un semáforo: éste puede tomar uno de tres valores: verde, amarillo o rojo.

Una posible codificación es emplear 1 bit (n=1) para indicar si el semáforo esta en rojo o no.

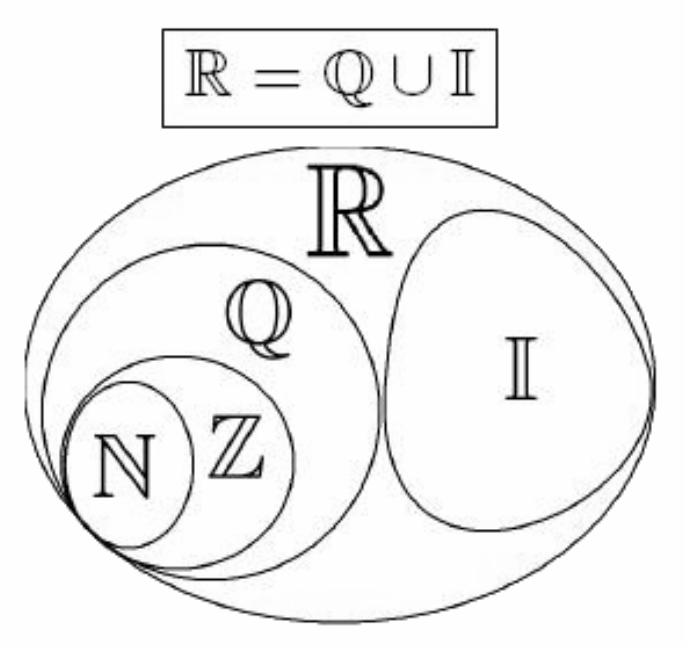
ROJO→1	NO ROJO→0	NO ROJO→0
		

Otra posibilidad es emplear 2 bits (n=2) para representar las tres situaciones posibles; ya que con dos bits pueden simbolizarse, en binario, cuatro casos.

ROJO→00	AMARILLO→01	VERDE→10
		

Existen 24 formas de representar los colores, ya que:
El código binario para el 1º color puede elegirse de 4 maneras diferentes,
El código binario para el 2º color puede elegirse de 3 maneras diferentes y
El código binario para el 3º color puede elegirse de 2 maneras diferentes.

Conjuntos Numéricos



- N: Naturales: $0,1,2,3,4,5...$
- Z: Enteros: $...-5,-4,-3,-2,-1,0,1,2,3,4,5...$
- Q: Racionales: $-\frac{5}{4},-\frac{8}{4},\frac{3}{2},\frac{7}{4},-\frac{8}{5},\frac{15}{5}...$
- I : Irracionales: $e=2,71...,\pi=3,14...,\sqrt{2}=1,41...$
- R: Reales: $-7, e=2,71...,\frac{15}{5},5,\pi=3,14...,2,\sqrt{2}=1,41...,-\frac{8}{4}...$

Códigos Binarios

Para La Representación De Números Naturales

- **Natural**

Se pueden representar números desde 0 hasta 2^{n-1}.

Si n=4

Decimal	Binario Natural
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Para La Representación De Números Enteros

- **Signo y Magnitud**

En esta codificación se reserva el bit más significativo para representar el signo del número y los restantes bits para representar su magnitud.
Se utiliza el "0" para el signo "+" y el "1" para el "-".

Se pueden representar números desde $-2^{n-1}+1$ hasta $2^{n-1}-1$.

- **Complemento a 1**

En esta codificación los números positivos se representan por su magnitud y los negativos por el complemento a 1 de su magnitud.

El complemento a 1 de una magnitud binaria se obtiene:
- Invirtiendo los unos por ceros y viceversa.

Se pueden representar números desde $-2^{n-1}+1$ hasta $2^{n-1}-1$.

- **Complemento a 2**

En esta codificación los números positivos se representan por su magnitud y los negativos por el complemento a 2 de su magnitud.

El complemento a 2 de una magnitud binaria se obtiene:

- Sumándole 1 al complemento a 1 de una magnitud.

- Analizando los bits de la magnitud a partir del menos significativo. Todos los bits encontrados hasta el primer uno inclusive tienen el mismo valor en la magnitud y su complemento, los bits hallados más allá de la izquierda del primer 1 aparecen invertidos en el complemento en relación a la magnitud.

Se pueden representar números desde -2^{n-1} hasta $2^{n-1}-1$.

Si n=4

Decimal	Signo y Magnitud	Complemento a 1	Complemento a 2
7	0111	0111	0111
6	0110	0110	0110
5	0101	0101	0101
4	0100	0100	0100
3	0011	0011	0011
2	0010	0010	0010
1	0001	0001	0001
0	0000 ó 1000	0000 ó 1000	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8			1000

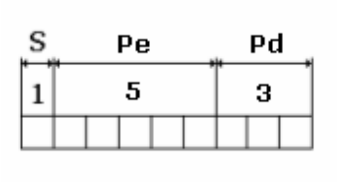
Para La Representación De Números Reales

- **Coma Fija**

$$(-1)^S \times P_e \ , \ P_d$$

- Signo (S): 0_ positivo, 1_negativo
- Parte Entera (Pe):
- Parte Decimal (Pd):

Si n = 9



$-10101{,}110 = (-1)^1 \times (1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3}) = -21{,}75_{10}$

$+01001{,}011 = (-1)^0 \times (0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}) = 9{,}375_{10}$

$+11111{,}111 = (-1)^1 \times (1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}) = 31{,}875_{10}$

$-11111{,}111 = (-1)^0 \times (1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}) = -31{,}875_{10}$

Decimal	Coma Fija
-21,75	110101110
9,375	001001011
31,875	111111111
-31,875	011111111

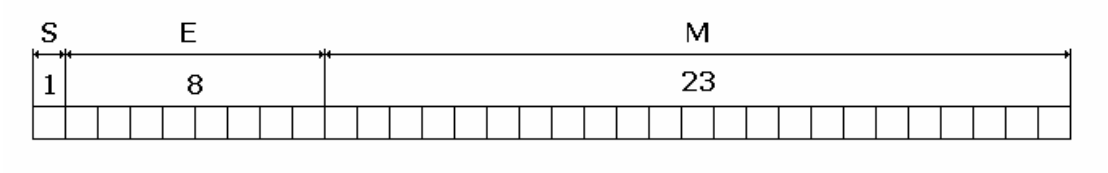
- **Coma Flotante**

$$(-1)^S \times M \times 2^E$$

- Signo (S): 0_ positivo, 1_negativo
- Mantisa (M):
- Exponente (E): Esta sesgado

Sesgo = $2^{k-1}-1$
k = número de bits del exponente.

Si n = 32



Decimal	Coma Flotante
7	01000000111000000000000000000000
-7	11000000111000000000000000000000
15	01000001011100000000000000000000
29	01000001111010000000000000000000
10,666...	01000001001010101010101010101010
0,666...	00111111001010101010101010101010

Para La Representación De Caracteres

- **ASCII**

Código Estadounidense Estándar para el Intercambio de Información. Creado en 1963 por el Comité Estadounidense de Estándares como mejora de los conjuntos de códigos utilizados en telegrafía.

n=8

Caracter	ASCII	Caracter	ASCII	Caracter	ASCII	Caracter	ASCII
NUL	00000000	Space	00100000	@	01000000	`	01100000
SOH	00000001	¡	00100001	A	01000001	a	01100001
STX	00000010	“	00100010	B	01000010	b	01100010
ETX	00000011	#	00100011	C	01000011	c	01100011
EOT	00000100	$	00100100	D	01000100	d	01100100
ENQ	00000101	%	00100101	E	01000101	e	01100101
ACK	00000110	&	00100110	F	01000110	f	01100110
Bell	00000111	‘	00100111	G	01000111	g	01100111
BS	00001000	(	00101000	H	01001000	h	01101000
HT	00001001	)	00101001	I	01001001	i	01101001
LF	00001010	*	00101010	J	01001010	j	01101010
VT	00001011	+	00101011	K	01001011	k	01101011
FF	00001100	,	00101100	L	01001100	l	01101100
CR	00001101	-	00101101	M	01001101	m	01101101
SO	00001110	.	00101110	N	01001110	n	01101110
SI	00001111	/	00101111	O	01001111	o	01101111
DEL	00010000	0	00110000	P	01010000	p	01110000
DC1	00010001	1	00110001	Q	01010001	q	01110001
DC2	00010010	2	00110010	R	01010010	r	01110010
DC3	00010011	3	00110011	S	01010011	s	01110011
DC4	00010100	4	00110100	T	01010100	t	01110100
NAK	00010101	5	00110101	U	01010101	u	01110101
SYN	00010110	6	00110110	V	01010110	v	01110110
ETB	00010111	7	00110111	W	01010111	w	01110111
CAN	00011000	8	00111000	X	01011000	x	01111000
EM	00011001	9	00111001	Y	01011001	y	01111001
SUB	00011010	:	00111010	Z	01011010	z	01111010
ESC	00011011	;	00111011	[	01011011	{	01111011
FS	00011100	<	00111100	\	01011100	|	01111100
GS	00011101	=	00111101	]	01011101	}	01111101
RS	00011110	>	00111110	^	01011110	~	01111110
US	00011111	?	00111111	_	01011111	DEL	01111111

- **ASCII Extendido**

En 1986, se modifico el estándar para agregar nuevos caracteres latinos, necesarios para la escrituras de textos en otros idiomas, como por ejemplo el español, así fue como se agregaron los caracteres que van del ASCII 128 al 255.

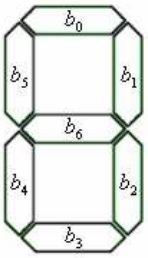
n=8

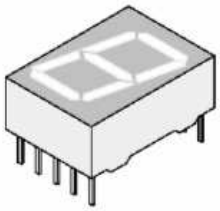
Caracter	ASCII	Caracter	ASCII	Caracter	ASCII	Caracter	ASCII
Ç	10000000	á	10100000	└	11000000	Ó	11100000
ü	10000001	í	10100001	┴	11000001	ß	11100001
é	10000010	ó	10100010	┬	11000010	Ô	11100010
â	10000011	ú	10100011	├	11000011	Ò	11100011
ä	10000100	ñ	10100100	─	11000100	õ	11100100
à	10000101	Ñ	10100101	┼	11000101	Õ	11100101
å	10000110	ª	10100110	ã	11000110	µ	11100110
ç	10000111	º	10100111	Ã	11000111	þ	11100111
ê	10001000	¿	10101000	╚	11001000	Þ	11101000
ë	10001001	®	10101001	╔	11001001	Ú	11101001
è	10001010	¬	10101010	╩	11001010	Û	11101010
ï	10001011	½	10101011	╦	11001011	Ù	11101011
î	10001100	¼	10101100	╠	11001100	ý	11101100
ì	10001101	¡	10101101	═	11001101	Ý	11101101
Ä	10001110	«	10101110	╬	11001110	¯	11101110
Å	10001111	»	10101111	¤	11001111	´	11101111
É	10010000	░	10110000	ð	11010000	≡	11110000
æ	10010001	▒	10110001	Ð	11010001	±	11110001
Æ	10010010	▓	10110010	Ê	11010010	‗	11110010
ô	10010011	│	10110011	Ë	11010011	¾	11110011
ö	10010100	┤	10110100	È	11010100	¶	11110100
ò	10010101	Á	10110101	ı	11010101	§	11110101
û	10010110	Â	10110110	Í	11010110	÷	11110110
ù	10010111	À	10110111	Î	11010111	¸	11110111
ÿ	10011000	©	10111000	Ï	11011000	°	11111000
Ö	10011001	╣	10111001	┘	11011001	¨	11111001
Ü	10011010	║	10111010	┌	11011010	•	11111010
ø	10011011	╗	10111011	█	11011011	¹	11111011
£	10011100	╝	10111100	▄	11011100	³	11111100
Ø	10011101	¢	10111101	¦	11011101	²	11111101
×	10011110	¥	10111110	Ì	11011110	■	11111110
ƒ	10011111	┐	10111111	▀	11011111	nbsp	11111111

De Despliegue

- **7 Segmentos**

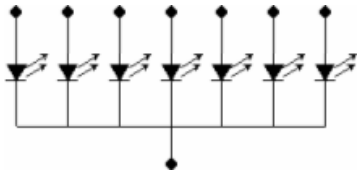
Código utilizado en los visualizadores de siete segmentos utilizados en equipos electrónicos como multímetros, relojes y calculadoras.





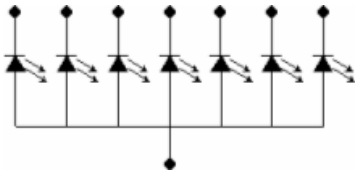
n=7

Estructura Cátodo Común



Decimal	7 Segmentos
0	0111111
1	0000110
2	1011011
3	1001111
4	1100110
5	1101101
6	1111101
7	0000111
8	1111111
9	1100111

Estructura Ánodo Común



Decimal	7 Segmentos
0	1000000
1	1111001
2	0100100
3	0110000
4	0011001
5	0010010
6	0000010
7	1111000
8	0000000
9	0011000

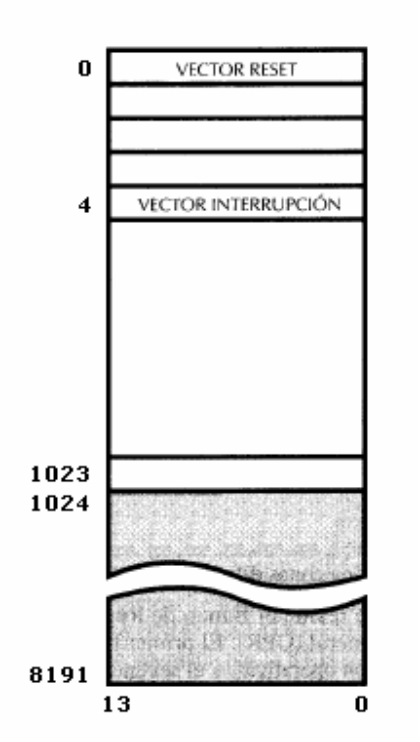
UNIDAD Nº3

Microcontrolador PIC16F84

Peripheral Interface Controller (Controlador de Interfaz Periférico)

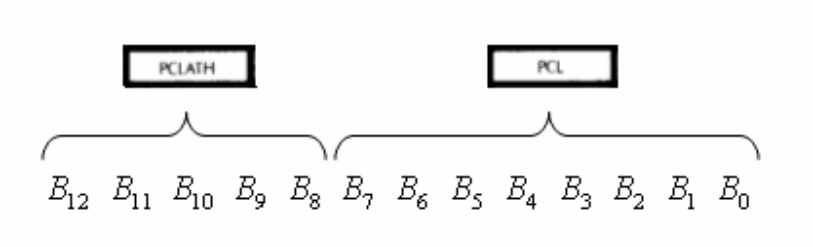
Memoria De Programa

Tipo Flash
1024x14

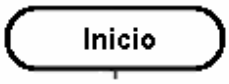
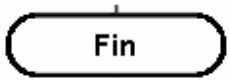
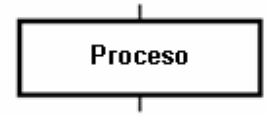
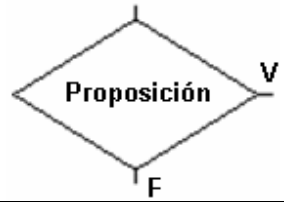


Contador De Programa

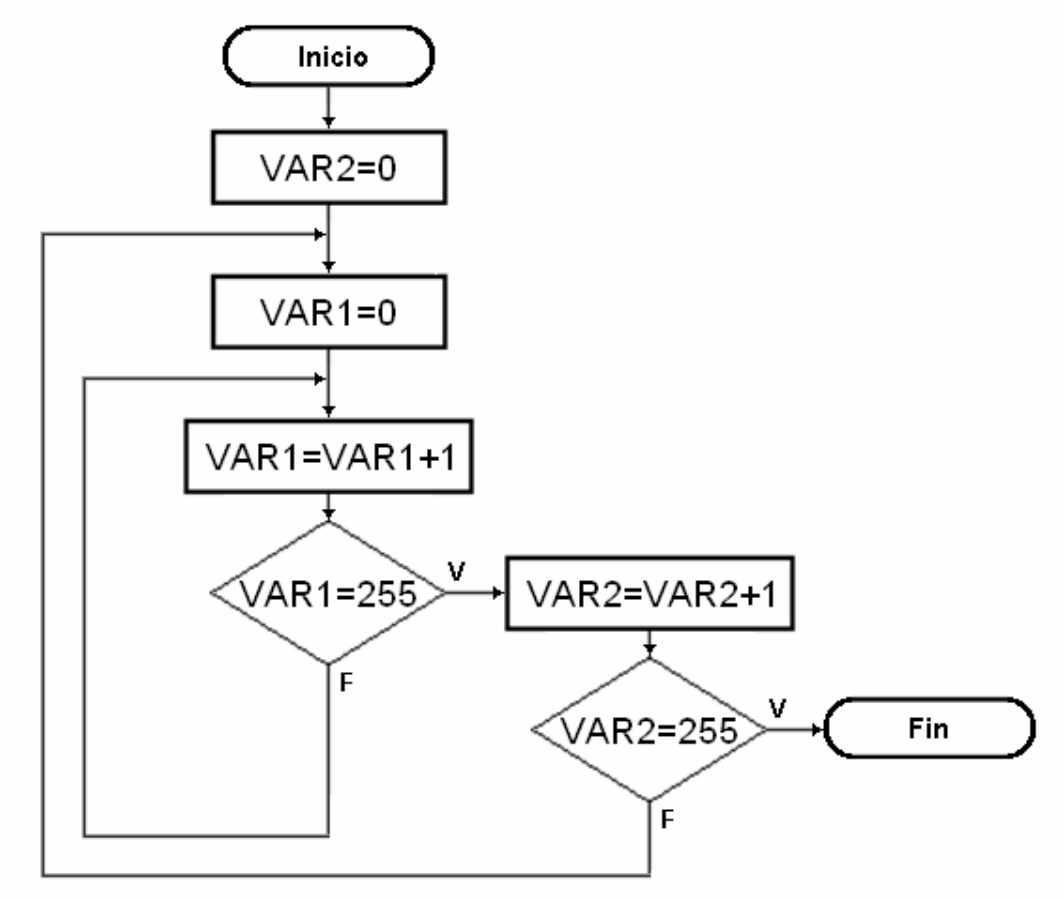
Contador de programa de 13 bits (permitiría direccionar 8192 posiciones de memoria, aunque el PIC solo tiene implementadas 1024 posiciones de memoria).



Representación Del Programa En Diagramas De Flujo

SIMBOLOS	
Grafico	**Representación**
	**Inicio del programa**
	**Fin del programa**
	**Proceso**
	**if (si)**

Ejemplo



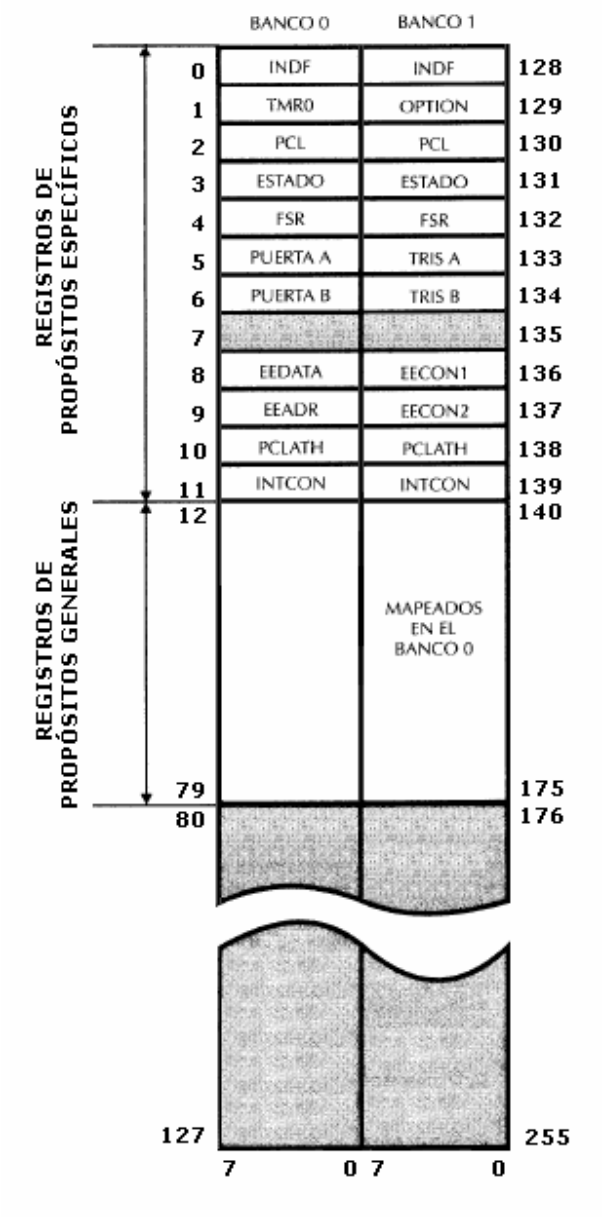
Set De Instrucciones

SINTAXIS	OPERACIÓN	CICLOS	FORMATO 14 bits	SEÑALIZADORES
INSTRUCCIONES QUE MANEJAN REGISTROS				
ADDWF f,d	Suma W y f	1	100 0111 dfff ffff	C,DC,Z
ANDWF f,d	AND entre W y f	1	00 0101 dfff ffff	Z
CLRF f	Pone todos los bits de f a 0	1	00 0001 1fff ffff	Z
CLRW ---	Pone todos los bits de W a 0	1	00 0001 0xxx xxxx	Z
COMF f,d	Complementa f (Invierte)	1	00 1001 dfff ffff	Z
DECF f,d	Decrementa f	1	00 0011 dfff ffff	Z
INCF f,d	Incrementa f	1	00 1010 dfff ffff	Z
IORWF f,d	OR entre W y f	1	00 0100 dfff ffff	Z
MOVF f,d	Mueve f	1	00 1000 dfff ffff	Z
MOVWF f	Mueve W a f	1	00 0000 1fff ffff	---
NOP ---	No opera	1	00 0000 0xx0 0000	---
RLF f,d	Rota f a la izqda. a través del acarreo	1	00 1101 dfff ffff	C
RRF f,d	Rota f a la dcha. a través del acarreo	1	00 1100 dfff ffff	C
SUBWF f,d	Resta W a f	1	00 0010 dfff ffff	C,DC,Z
SWAPF f,d	Intercambia nibbles	1	00 1110 dfff ffff	---
XORWF f,d	XOR entre W y f	1	00 0110 dfff ffff	Z
INSTRUCCIONES QUE MANEJAN BITS				
BCF f,b	Pone a 0 el bit b de f	1	01 00bb bfff ffff	---
BSF f,b	Pone a 1 el bit b de f	1	01 01bb bfff ffff	---
INSTRUCCIONES DE "BRINCO"				
BTFSC f,b	Explora el bit b de f y si vale 0, brinca	1(2)	01 10bb bfff ffff	---
BTFSS f,b	Explora el bit b de f y si vale 1, brinca	1(2)	01 11bb bfff ffff	---
DECFZF f,d	Decrementa f y si es 00000000, brinca	1(2)	00 1011 dfff ffff	---
INCFSZ f,d	Incrementa f y si es 11111111, brinca	1(2)	00 1111 dfff ffff	---
INSTRUCCIONES QUE MANEJAN OPERANDOS INMEDIATOS				
ADDLW k	Suma inmediata con W	1	11 111x kkkk kkkk	C.DC,Z
ANDLW k	AND inmediato con W	1	11 1001 kkkk kkkk	Z
IORLW k	OR inmediato con W	1	11 1000 kkkk kkkk	Z
MOVLW k	Mueve a W un valor inmediato	1	11 00xx kkkk kkkk	---
SUBLW k	Resta W de un inmediato	1	11 110x kkkk kkkk	C.DC,Z
XORLW k	OR exclusiva con W	1	11 1010 kkkk kkkk	Z
INSTRUCCIONES DE CONTROL Y ESPECIALES				
CALL e	Llamada a subrutina	2	10 0eee eeee eeee	TO#,PD#
CLRWDT	Borra o refresca el Perro Guardián	1	00 0000 0110 0100	---
GOTO e	Salto incondicional	2	10 1eee eeee eeee	---
RETFIE	Retorno de Interrupción (GIE=1)	2	00 0000 0000 (00)	---
RETLW k	Retorno subrutina y carga W=k	2	11 01xx kkkk kkkk	---
RETURN	Retorno de subrutina	2	00 0000 0000 01000	---
SLEEP	Pasa al modo de reposo	1	00 0000 0110 0011	TOP#,PD#

f	file = nombre (cadena de caracteres ASCII) del registro
d	destiny = destino (d=0 → W; d=1 → f)
b	bit (0,1,2,3,4,5,6,7)
k	constante = número (0,1,2,3,4,5 … 255)
e	etiqueta = cadena de caracteres ASCII

Memoria De Datos

Tipo SRAM
128x8



Registros De Propósitos Específicos

DIR.	NOMBRE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	VALOR OTRAS EL RESET	VALOR OTROS RESET
BANCO 0											
00 h	INDF	Usa el contenido de FSR para direccionar la memoria (no es un registro físico)								---- ----	---- ----
01 h	TMR0	Reloj/Contador en tiempo real de 8 bits								xxxx xxxx	uuuu uuuu
02 h	PCL	8 bits menos significativos del PC								0000 0000	0000 0000
03 h	ESTADO	IRP	RP1	RP0	TO#	PD#	Z	DC	C	0001 1xxx	000q quuu
04 h	FSR	Direccionamiento indirecto con INDF								xxxx xxxx	uuuu uuuu
05 h	PUERTA A	---	---	---	RA4/TOCK	RA3	RA2	RA1	RA0	---x xxxx	---u uuuu
06 h	PUERTA B	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	uuuu uuuu
07 h		No Implementado, se lee como "0"								--- ---	--- ---
08 h	EEDATA	Registro de datos de EEPROM								xxxx xxxx	uuuu uuuu
09 h	EEADR	Registro de direcciones de EEPROM								xxxx xxxx	uuuu uuuu
0A h	PCLATH	---	---	---	5 bits mas significativos del PC					---0 0000	---0 0000
0B h	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
BANCO 1											
80 h	INDF	Usa el contenido de FSR para direccionar la memoria (no es un registro físico)								---- ----	---- ----
81 h	OPTION	RBPU#	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
82 h	PCL	8 bits menos significativos del PC								0000 0000	0000 0000
83 h	ESTADO	IRP	RP1	RP0	TO#	PD#	Z	DC	C	0001 1xxx	000q quuu
84 h	FSR	Direccionamiento indirecto con INDF								xxxx xxxx	uuuu uuuu
85 h	TRISA	---	---	---	Configuración Puerta A					---1 1111	---1 1111
86 h	TRISB	Configuración Puerta B								1111 1111	1111 1111
87 h		No complementado, se lee "0"								---- ----	---- ----
88 h	EECON1	---	---	---	EEIF	WRERR	WREN	WR	RD	---0 x000	---- q000
89 h	EECON2	Registro de Control EEPROM (no es un registro físico)								---- ----	---- ----
8A h	PCLATH	---	---	---	5 bits mas significativos del PC					---0 0000	---0 0000
8B h	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u

x	desconocido
u	no cambia
–	no implementado, se lee como 0
q	valor que depende de la condición

BANCO 0

- INDF (direccionamiento indirecto): Dirección 00h, sirve para ver el dato de la dirección a la que apunta el registro FSR (dir. 04h) que veremos mas adelante
- TMR0 (Timer/contador): Dirección 01h, Aquí se puede ver el valor en tiempo real del Timer/contador. También se puede introducir un valor y alterar así el conteo. Este conteo puede ser interno (cuenta ciclos de reloj) o externo (cuneta pulsos introducidos por RA4).
- PCL (Parte baja del contador de programa): Dirección 02h, Modificando este registro se modifica el contador de programa, este contador de programa es el que señala al PIC en que dirección (de EEPROM) tiene que leer la siguiente instrucción. Esto se utiliza mucho para consultar tablas (ya veremos mas adelante)
- STATUS: Dirección 03h, este es uno de los registros mas importantes y el que mas vas a utilizar. Hay que analizar el funcionamiento de este registro bit a bit:
    - CARRY, Dirección STATUS, 0 (bit 0): bit de desbordamiento. Este bit se pone a "1" cuando la operación anterior ha rebasado la capacidad de un byte. Por ejemplo, si sumo dos números y el resultado no cabe en 8 bit el CARRY se pone a "1", Pasa lo mismo cuando resto dos números y el resultado es un número negativo. Se puede usar para saber si un número es mayor que otro (restándolos, si hay acarreo es que el segundo era mayor que el primero). Una vez que este bit se pone a "1" no se baja solo (a"0"), hay que hacerlo por programa si queremos volverlo a utilizar.
    - DC (digit carry), Dirección STATUS,1 (bit 1): lo mismo que el anterior pero esta vez nos avisa si el número no cabe en cuatro bits.
    - Z (zero), Dirección STATUS,2 (bit 2): Se pone a "1" si la operación anterior ha sido cero. Y pasa a "0" si la operación anterior no ha sido cero. Se usa para comprobar la igualdad entre dos números (restándolos, si el resultado es cero ambos números son iguales)
    - PD (Power - Down bit), Dirección STATUS,3 (bit3) se pone a "0" después de ejecutar la instrucción SLEEP*, se pone a "1" después de ejecutar la instrucción CLRWDT* o después de un power-up*.
    - TO (Timer Up), Dirección STATUS,4 (bit4) se pone a "0" cuando se acaba el tiempo del WATCHDOG*, Se pone a "1" después de ejecutar las instrucciones, CLRWDT* o SLEEP* o después de un power-up*.
    - RP0 y RP1 (selección de banco), Dirección STATUS,5 y STATUS,6. Como el PIC16F84 solo tiene dos bancos de memoria el RP1 no se usa para nada, la selección del banco se hace mediante RP0 (STATUS,5), si está a "0" nos encontramos en el banco 0, y si está a "1" nos encontramos en el banco 1.
    - IRP, Dirección STATUS,7, En este PIC no se usa para nada.

- FSR (Puntero), Dirección 04h, se usa para direccionamiento indirecto en combinación con el registro INDF (dir. 00h): se carga la dirección del registro que queremos leer indirectamente en FSR y se lee el contenido de dicho registro en INDF.

- PORTA (Puerto A), Dirección 05h. Con este registro se puede ver o modificar el estado de los pines del puerto A (RA0 - RA4). Si un bit de este registro está a "1" también lo estará el pin correspondiente a ese bit. El que un pin esté a "1" quiere decir que su tensión es de 5V, si está a "0" su tensión es 0V.

  Correspondencia:
  - RA0 → PORTA,0
  - RA1 → PORTA,1
  - RA2 → PORTA,2
  - RA3 → PORTA,3
  - RA4 → PORTA,4

- PORTB (Puerto B), Dirección 06h igual que PORTA pero con el puerto B

  Correspondencia:
  - RB0 → PORTB,0
  - RB1 → PORTB,1
  - RB2 → PORTB,2
  - RB3 → PORTB,3
  - RB4 → PORTB,4
  - RB5 → PORTB,5
  - RB6 → PORTB,6
  - RB7 → PORTB,7

- Dirección 07h, No utilizada por este PIC.
- EEDATA, Dirección 08h. En este registro se pone el dato que se quiere grabar en la EEPROM de datos
- EEADR, Dirección 09h. En este registro se pone la dirección de la EEPROM de datos donde queremos almacenar el contenido de EEDATA
- PCLATH, Dirección 0Ah. Modifica la parte alta del contador de programa (PC), el contador de programa se compone de 13 bits, los 8 bits de menor peso se pueden modificar con PCL (dir. 02h) y los 5 bits de mayor peso se pueden modificar con PCLATH
- INTCON (controla las interrupciones), Dirección 0Bh. Se estudia bit a bit:
  - RBIF (Flag de interrupción por cambio de PORTB) Dirección INTCON,0 (bit 0) se pone a "1" cuando alguno de los pines RB4, RB5, RB6, o RB7 cambia su estado. Una vez que está a "1" no pasa a "0" por si mismo: hay que ponerlo a cero por programa.
  - INTF (Flag de interrupción de RB0) Dirección INTCON,1. Si está a "1" es que ha ocurrido una interrupción por RB0, si está a "0" es que dicha interrupción no ha ocurrido. Este bit es una copia de RB0.
  - TOIF (Flag de interrupción por desbordamiento de TMR0) Dirección INTCON,2. Cuando TMR0 se desborda este Flag avisa poniéndose a "1". Poner a "0" por programa.
  - RBIE (Habilita la interrupción por cambio de PORTB) Dirección INTCON,3. Si está a "1" las interrupciones por cambio de PORTB son posibles.

- INTE (Habilita la interrupción por RB0) Dirección INTCON,4. Si lo ponemos a "1" la interrupción por RB0 es posible.
- TOIE (Habilita la interrupción por desbordamiento de TMR0) Dirección INTCON,5. Si este bit esta a "1" la interrupción por desbordamiento de TMR0 es posible.
- EEIE (Habilita la interrupción por fin de escritura en la EEPROM de datos) Dirección INTCON,6. Cuando este bit está a "1" la interrupción cuando acaba la escritura en la EEPROM de datos es posible.
- GIE (Habilita las interrupciones globalmente) Dirección INTCON,7. Este bit permite que cualquier interrupción de las anteriores sea posible. Para usar alguna de las interrupciones anteriores hay que habilitarlas globalmente e individualmente.

Ahora vamos con el banco 1, solo un comentario antes de empezar: ¿recuerdas la tabla de registros internos que veíamos en el punto 2.2? (ver tabla) ves que los registros del banco 0 y los del banco 1 tienen direcciones distintas, en realidad podemos utilizar las mismas direcciones para referirnos a registros que están en uno u otro banco. El PIC las diferenciará sin problemas gracias al bit de selección de banco (RP0). Por ejemplo, la dirección 05h se refiere a PORTA si estamos en el banco 0 y a TRISA si estamos en el banco 2.

BANCO 1

- INDF, Dirección 00h, Igual que en el Banco 0
- OPTION, Dirección 01h, (configuración del prescaler, Timer, y alguna cosa mas) Se estudia bit a bit
  - PS0, PS1 y PS2 (Bits del 0 al 2) Configuración del preescaler: El preescaler es un divisor de pulsos que está a la entrada del Timer-contador. El prescaler divide el número de pulsos que le entran al timer-contador o al Watchdog. El factor de división es el siguiente (según los valores de PS2, PS1 y PS0 respectivamente

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

  - PSA, Dirección OPTION,3. Bit de asignación de prescaler: si está a "1" el prescaler se asigna a WDT (Watchdog), si está a "0" se asigna al TMR0

    - TOSE, Dirección OPTION,4. Bit de selección del tipo de flanco para TMR0. A "1" se incrementa TMR0 por flanco descendente de RA4, a "0" se incrementa TMR0 por flanco ascendente de RA4.
    - TOCS, Dirección OPTION,5. Selecciona la entrada de reloj de TMR0. A "1" la entrada de reloj de TMR0 es por flanco de la patilla RA4, a "0" la entrada de reloj de TMR0 es por ciclo de reloj interno.
    - INTEDG, Dirección OPTION,6. Tipo de flanco para la interrupción por RB0: A "1" la interrupción será por flanco ascendente, a "0" la interrupción será por flanco descenderte.
    - RBPU, dirección OPTION,7. Carga Pull-Up en puerto B. A "0" todas las salidas del puerto B tendrán una carga de pull-Up interna.

- PCL, Dirección 02h, igual que en el banco 0
- STATUS, Dirección 03h, Igual que en el banco 0
- FSR, Dirección 04h, Igual que en el banco 0
- TRISA, Dirección 05h, Configura el puerto A como entrada o salida. Si un bit de este registro se pone a "0" el pin correspondiente en el puerto A será una salida, por el contrario, si se pone a "1" el pin correspondiente en el puerto A será una entrada.
- TRISB, Dirección 06h, Igual que el anterior pero con el puerto B
- Dirección 07h, No usada en este PIC
- EECON1, Dirección 08h, Controla la lectura y escritura en la EEPROM de datos. Se estudia bit a bit:
    - RD, Dirección EECON1,0 (bit 0) A "1" iniciamos el ciclo de lectura, cuando acaba el ciclo se pone a "0" el solito
    - WR, Dirección EECON1,1 (bit 1) A "1" indicamos que comienza el ciclo de escritura, cuando acaba el ciclo se pone a "0" él solito
    - WREN, Dirección EECON1,2 (bit 2) si lo ponemos a "1" se permite la escritura, a "0" no se permite.
    - WRERR, Dirección EECON1,3 (bit 3) error de escritura, si está a "1" indica que no se ha terminado el ciclo de escritura.
    - EEIF, Dirección EECON1,4 (bit 4) interrupción de ciclo de escritura de la EEPROM, si está a "1" indica que el ciclo de escritura ha terminado, hay que ponerlo a "0" por programa.
    - Bits del 5 al 7 no se utilizan.

- EECON2, Dirección 09h, Se utiliza para la escritura en la EEPROM de datos como medida de seguridad: para poder guardar algo en la EEPROM hay que cargar el valor 55h en este registro.
- PCLATH, Dirección 0Ah, Igual que en el banco 0
- INTCON, Dirección 0Bh, Igual que en el banco 1

Directivas Del Ensamblador

ORG

Define el origen del programa, por definición del fabricante se ubica en la posición de memoria 0, debido al problema al problema de los vectores a continuación se debe realizar un salto donde comenzará a ejecutarse el programa principal. Luego se debe definir el origen de las interrupciones, realizando también un salto, por ultimo se indica el origen donde comenzará a ejecutarse el programa principal.

Ejemplo

```
ORG 0H
GOTO INICIO
ORG 4H
GOTO INTERRUPCION
ORG 6H
```

EQU

Instrucción que permite definir los registros, primero se define el nombre y a continuación la posición de memoria que le corresponde.

Ejemplo

```
VAR1 EQU 10H
PORTA EQU 5H
ESTADO EQU 3H
TRISB EQU 86H
SUMA EQU 2CH
```

END

Indica al ensamblador el fin del programa, debe ser la última instrucción, no lleva punto y coma y no se puede colocar otra instrucción en la misma línea.

Codificación Del Programa En Ensamblador

Modelo de microcontrolador

```
LIST        P=16F84
```

Configuración de fusibles

```
CP_ON       equ     0x000F ;    Activa code protect
CP_OFF      equ     0x3FFF ;    Desactiva code protect
PWRTE_ON    equ     0x3FF7 ;    Activa power on reset
PWRTE_OFF   equ     0x3FFF ;    Desactiva power on reset
WDT_ON      equ     0x3FFF ;    Activa Watchdog
WDT_OFF     equ     0x3FFB ;    Desactiva Watchdog
LP_OSC      equ     0x3FFC ;    Oscilador LP
XT_OSC      equ     0x3FFD ;    Oscilador XT
HS_OSC      equ     0x3FFE ;    Oscilador HS
RC_OSC      equ     0x3FFF ;    Oscilador RC

__CONFIG CP_OFF & WDT_OFF & RC_OSC & PWRTE_ON
```

Declaración de registros

```
[Etiqueta]     equ   [Dirección De Memoria]   [;Comentario]

Estado1  equ   0x03   ;  Registros De Propósitos Específicos
Puertob  equ   0x06   ;  <Banco 0>

Estado2  equ   0x83   ;  Registros De Propósitos Específicos
Trisb    equ   0x86   ;  <Banco 1>

Var1     equ   0x0C   ;  Registros De Propósitos Generales
Var2     equ   0x0D   ;
Var3     equ   0x0E   ;
```

Inicio del programa

```
org     0x00
goto    inicio
org     0x05
inicio
```

Cuerpo del programa

```
[Etiqueta]   Comando   [Operando/s]   [;Comentario]

         bsf     Estado1,5       ;       pone RP0 a 1 y pasa al Banco1
         clrf    Trisb           ;       pone todos los bits de Trisb a 0
         bcf     Estado2,5       ;       pone RP0 a 0 y pasa al Banco0
 repetir bsf     Puertob,0       ;       pone a 1 RB0
         bcf     Puertob,0       ;       pone a 0 RB0
         goto    repetir         ;       se repiten las ultimas 2 instrucciones
```

Fin del programa

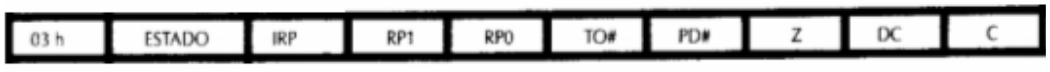
```
end
```

Representación De Números

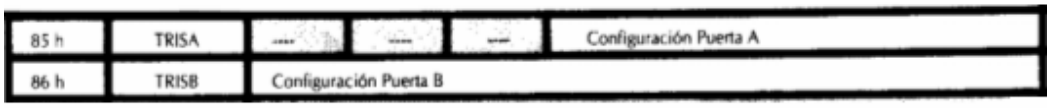
TIPO	SINTAXIS	EJEMPLO
Decimal	D'<cantidad>' d'<cantidad>' .<cantidad>	D'109' d'109' .109
Hexadecimal	H'<cantidad>' h'<cantidad>' 0x<cantidad> <cantidad>H <cantidad>h	H'6D' h'6D' 0x6D 6DH 6Dh
Octal	O'<cantidad>' o'<cantidad>'	O'155' o'155'
Binario	B'<cantidad>' b'<cantidad>'	B'01101101' b'01101101'
ASCII	A'<carácter>' a'<carácter>' '<carácter>'	A'M' a'M' 'M'

Manejo de Puertos

Para configurar la dirección del flujo de la información (entrada o salida) de los puertos existen los registros trisA y trisB, cada bit de estos registros indica la dirección (0 para salida, 1 para entrada) de los pines al que corresponden, (trisA → puerto A y trisB → puerto B). Inicialmente el µC se encuentra ubicado en el banco 0 pero los registros trisA y trisB se encuentran en el banco 1, por lo tanto antes de programarlos se debe pasar al banco 1 utilizando el bit 5 del registro de estados(0 para el banco 0, 1 para el banco 1).



RP0 = 0 → Banco 0
RP0 = 1 → Banco 1



TRISA = '11111' → PUERTA A ENTRADA; TRISA = '00000' → PUERTA A SALIDA

TRISB = '11111111' → PUERTA B ENTRADA; TRISB = '00000000' → PUERTA B SALIDA

Control Del Flujo

Para controlar el flujo de ejecución del programa se utilizan los saltos.

Saltos Condicionales

- **BTFSC**
- **BTFSS**

Realizan el salto en base al estado en que se encuentra un determinado bit de un registro. En el caso que la condición de salto sea afirmativa el PC salta una instrucción, en el caso de ser negativa continua con el programa.

- **DECFSZ**
- **INCFSZ**

Realizan el salto si un registro supera el número 255 o si llegue a 0, en ambos casos recién se realiza el salto, si no se cumple continua con el programa.

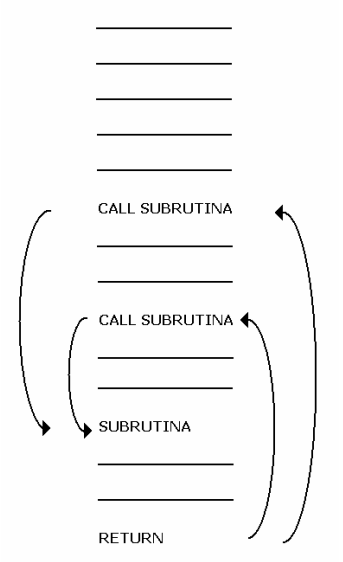
Saltos Incondicionales

- **GOTO**

Realiza el salto a donde se encuentre la etiqueta dentro de programa. Como en los saltos condicionales solo se realiza el salto de una instrucción en caso de ser la respuesta negativa, se utiliza esta instrucción para indicar donde se va a realizar la opción en caso de ser negativa.

- **CALL**
- **RETURN**

Realiza el salto y retorno de las subrutinas. Una subrutina es una parte de programa que hace algo concreto y se repite a menudo, para ahorrar memoria y esfuerzo y para hacer el programa mas comprensible se agrupa en forma de subrutina. Una subrutina se debe ejecutar siempre llamándola con la instrucción CALL y al final de dicha subrutina debe haber siempre un RETURN.
Durante el programa principal se llama varias veces a la subrutina SUBR (el nombre es lo de menos) con la instrucción CALL. Cuando el PIC ejecuta una instrucción CALL se guarda en memoria la dirección de código de programa a la que tiene que retornar de tal forma que cuando se encuentra con la instrucción RETURN vuelve al programa principal donde lo dejó.
Una subrutina puede ser llamada desde el programa principal, también desde otra subrutina o desde una interrupción.

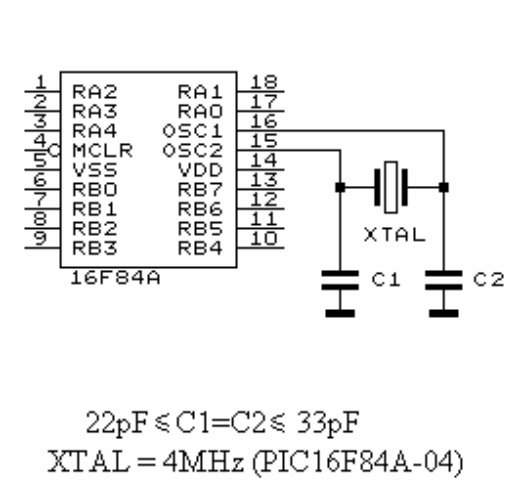


Los Fusibles

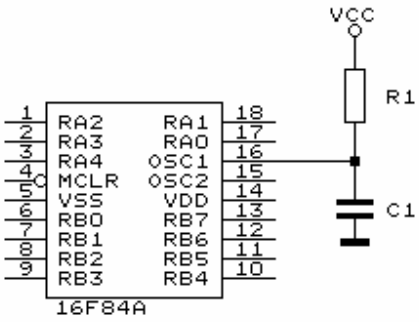
Configuran ciertos aspectos de funcionamiento cuando se graba el PIC.

- **OSC (Oscillator – Oscilador)**
  Es el tipo de oscilador que va a utilizar el PIC:

- XT: Cristal de cuarzo y dos capacitores.
- RC: Resistencia y capacitor.



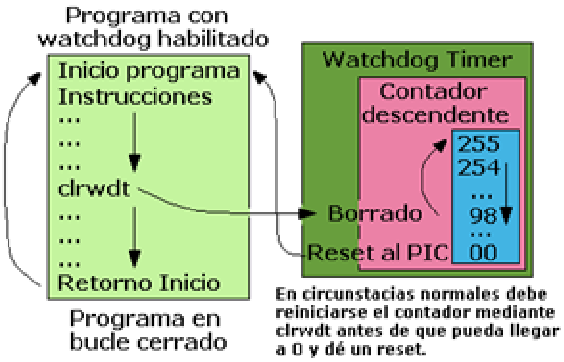
$22pF \leq C1=C2 \leq 33pF$
$XTAL = 4MHz\ (PIC16F84A\text{-}04)$



$C1 \geq 20pF$
$5K\Omega > R1 > 100K\Omega$

- **WDT (Watchdog Timer – Temporizador Perro Guardián)**

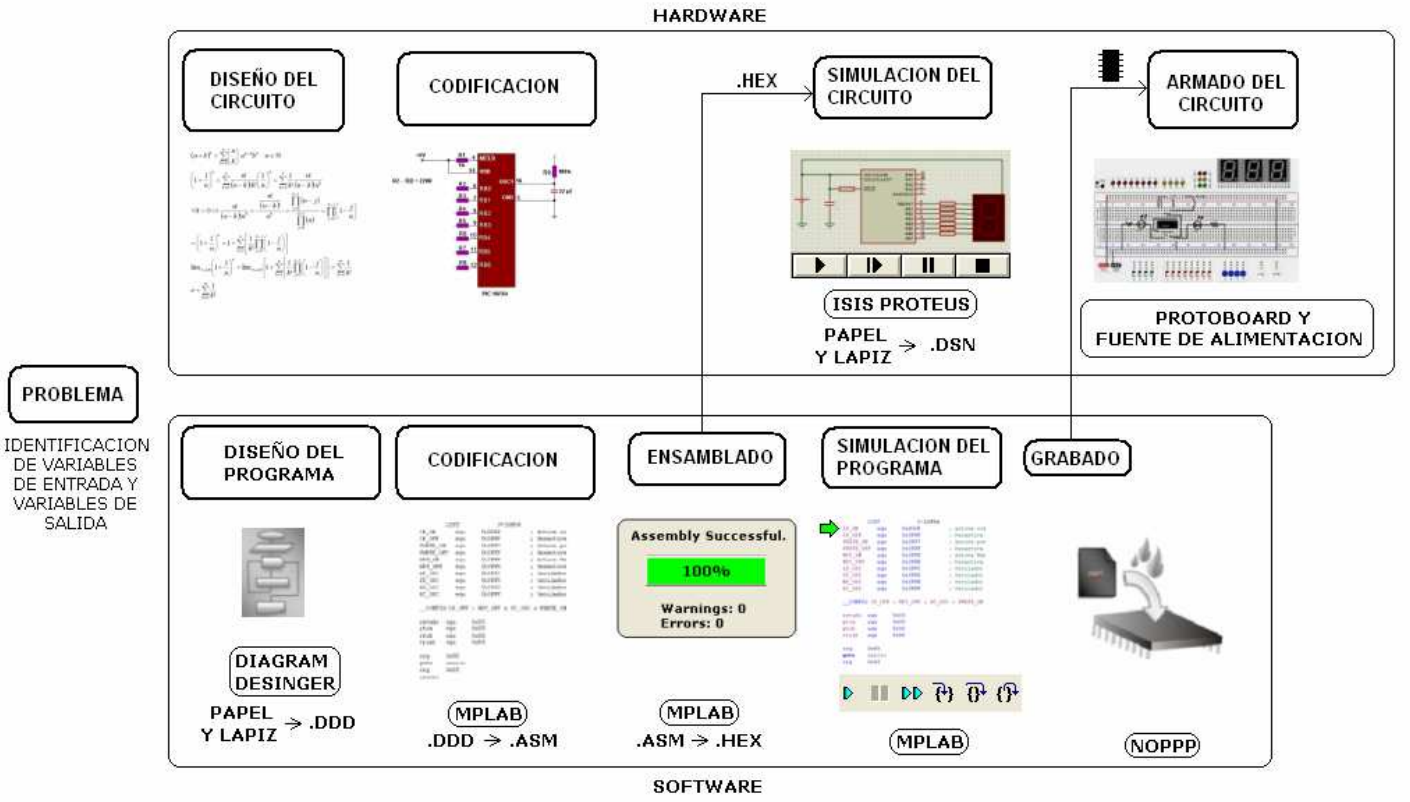
  Permite que el PIC se resetee automáticamente si por un descuido de programación, se queda en un bucle infinito. Para utilizarlo debemos borrar cada cierto tiempo el registro del Perro Guardián. Si transcurrida la ejecución de una cierta cantidad de instrucciones el registro no ha sido borrado el PIC se resetea. La instrucción para borrar el registro es CLRWDT.



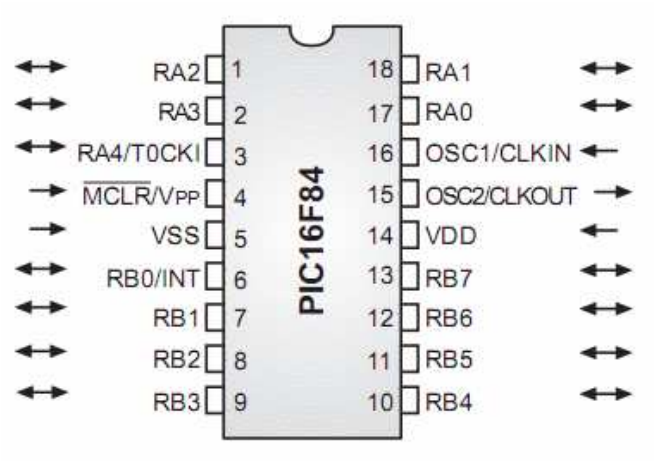
- **PWRT (Power Up Timer Reset –Temporizador De Reseteo De Encendido)**
  Al activarlo se genera un retardo en la inicialización del PIC. Esto se usa para esperar que la tensión se estabilice.

- **CP (Code Protect - Protección Del Código)**
  Impide leer y sobrescribir el contenido del PIC.

Etapas De Diseño



Distribución De Los Terminales



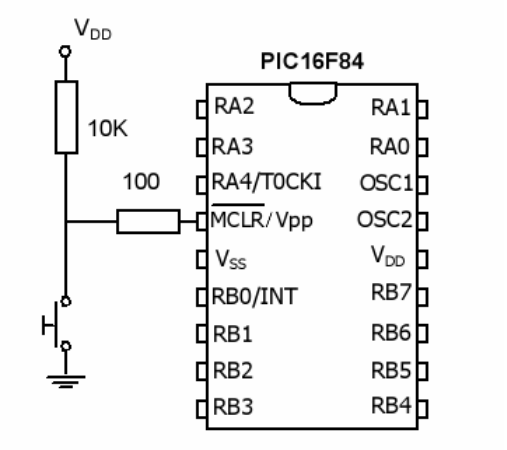
- **V_{DD} y V_{SS} :** Alimentación (2 V - 6 V).
- **$\overline{MCLR}$** (*Master Clear*): Reset cuando la tensión < 1,7V .
- **RA0- RA4:** Entradas/salidas del puerto A.
  - I_O = 20mA; $\sum I_O$ = 50mA
  - I_I = 25mA; $\sum I_I$ = 80mA
  - RA4/T0CK1 puede ser contador/temporizador externo TMR0.
- **RB0-RB7:** Entradas/salidas del puerto B.
  - I_O = 20mA; $\sum I_O$ = 100mA
  - I_I = 25mA; $\sum I_I$ = 150mA
  - RB0/INT puede ser interrupción externa.
- **OSC1/CLKIN OSC2/CLKOUT :** conexión del oscilador del reloj principal.

MCLR (Master Clear)

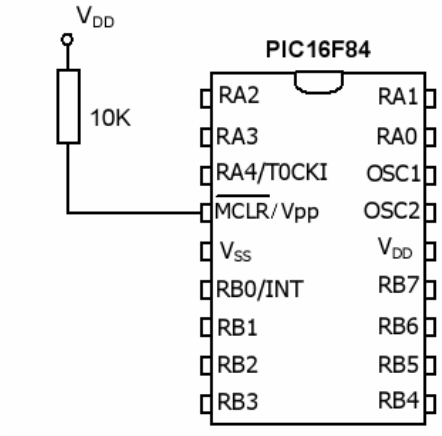
Pin de Reseteo del Microcontrolador.
Se activa cuando tiene un "0" lógico en su entrada.

Si se utiliza

Activado con pulsador



Si no se utiliza



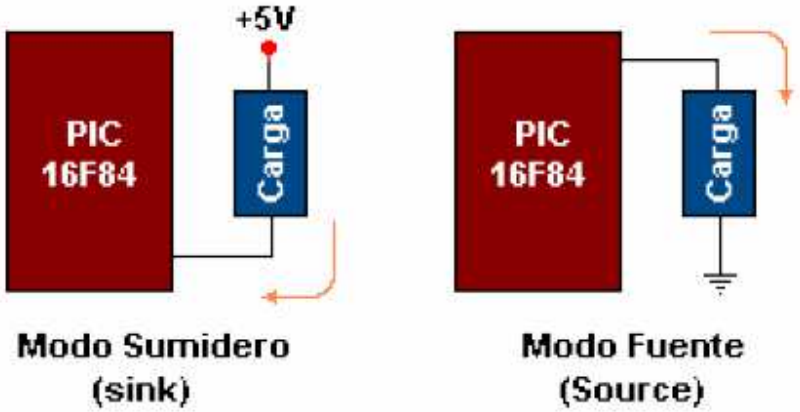
PORTA — 05h Banco 0

Puerto de entrada/salida de 5 bits (pines RA0 a RA4). El puerto A puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de sus pines se llama TRISA y esta localizado en la dirección 85h del Banco 1. Su pin RA4/TOCKI también puede servir de entrada al Timer 0. Al conectar la alimentación queda configurado como entrada.

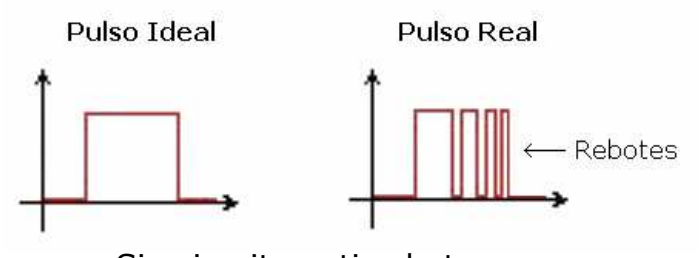
PORTB — 06h Banco 0

Puerto de entrada/salida de 8 bits (pines RB0 a RB7). El puerto B puede leerse o escribirse como si se tratara de un registro cualquiera. El registro que controla el sentido (entrada o salida) de sus pines se llama TRISB y está localizado en la dirección 86h del Banco 1. Algunos de sus pines tienen funciones alternas en la generación de interrupciones. Al conectar la alimentación queda configurado como entrada.

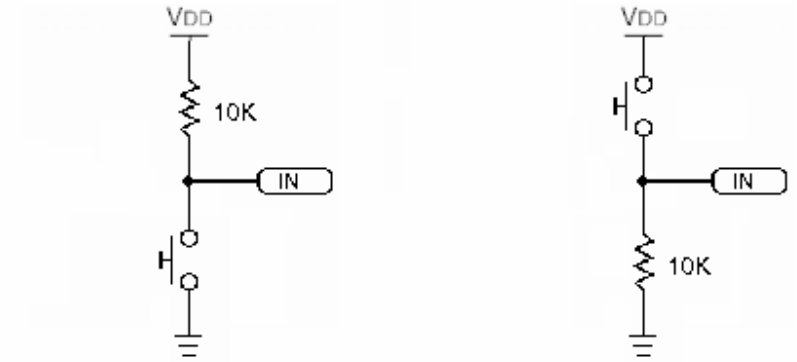
	PUERTO A	PUERTO B
Modo Sumidero	80 mA	150 mA
Modo Fuente	50 mA	100 mA



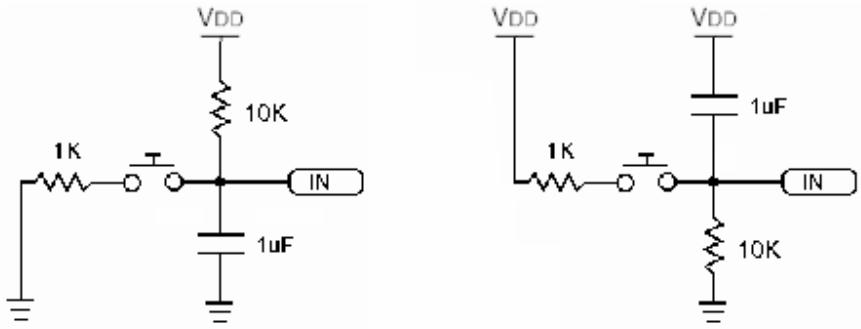
Circuitos de prueba para entradas



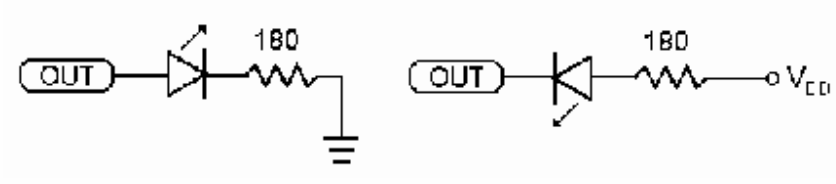
Sin circuito antirrebote



Con circuito antirrobote



Circuitos de prueba para salidas



1.

Configure → Select Device → PIC16F84

2.

Debugger → Select tool → MPLAB SIM

3.

New → Escribir el programa → Guardarlo con extensión .ASM

4.

Project → Quickbuild

Simulación

View →

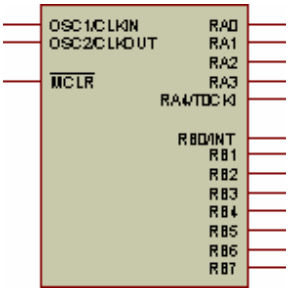
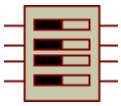
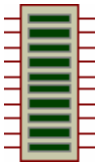
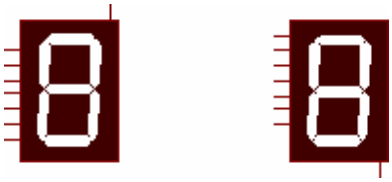
Memoria De Datos
1 File register

Memoria De Programa
2 Program memory

Reset → Stef info (paso a paso)

PROTEUS DESIGN SUITE
ISIS SCHEMATIC CAPTURE

Component Mode

	**PIC16F84A**
	**BUTTON**
	**DIPSW_2** **...** **DIPSW_10**
	RES
	**RES-VAR**
	**LED-BLUE** **LED-GREEN** **LED-RED** **LED-YELLOW**
	**LED-BARGRAPH-GRN**
	**7SEG-COM-ANODE** **7SEG-COM-CATHODE**
	**CAP**
	**CRYSTAL**

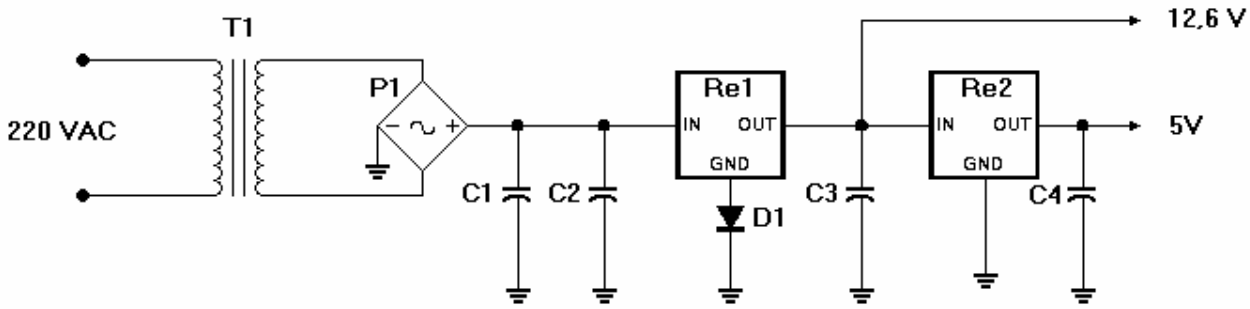
Terminals Mode

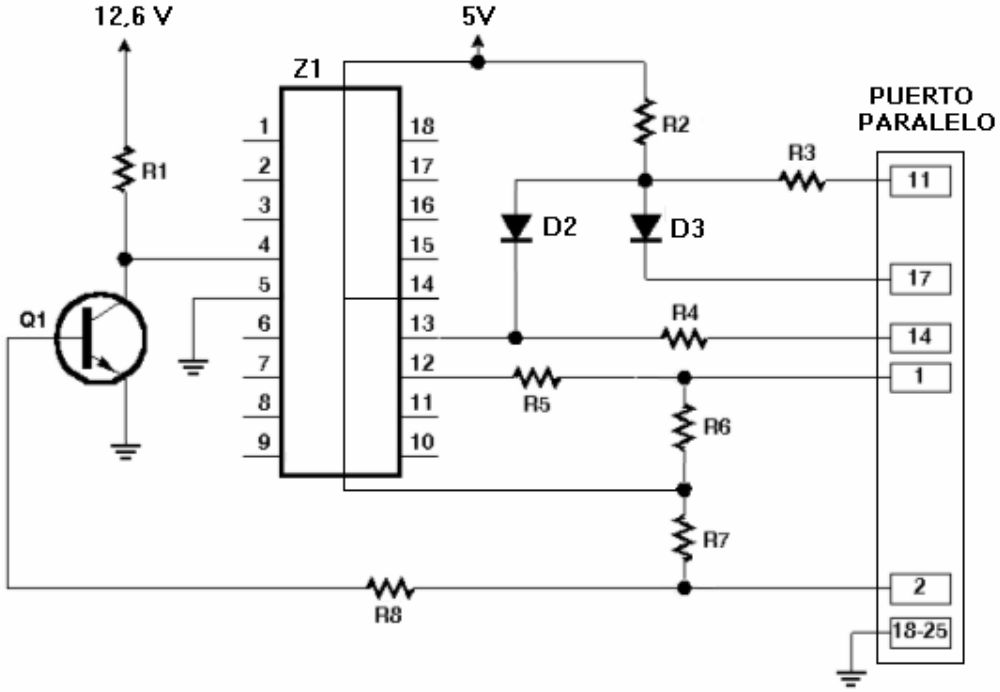
	**GROUND**

Grabador De PIC

- 16C84
- 16F83
- 16F84

Circuito Eléctrico





Componentes

- C1 = 1 μF x 36 V
- C2 = 2200 μF x 36 V
- C3 = 0,1 μF x 36 V
- C4 = 0,1 μF x 36 V

- R1 = 2,2 $K\Omega$ x ¼ W
- R2 = 4,7 $K\Omega$ x ¼ W
- R3 = 330 Ω x ¼ W
- R4 = 1 $K\Omega$ x ¼ W
- R5 = 1 $K\Omega$ x ¼ W
- R6 = 4,7 $K\Omega$ x ¼ W
- R7 = 4,7 $K\Omega$ x ¼ W
- R8 = 1 $K\Omega$ x ¼ W

- Q1 = 2N3904
- D1 = 1N4148
- D2 = 1N60
- D3 = 1N60
- P1 = Puente rectificador de diodos

- T1 = Transformador de 12V x 20mA

- Re1 = 7812
- Re2 = 7805

- Z1 = Zócalo para integrado de 18 pines

Sustituciones

Re1 = 78M12 = 78L12
Re2 = 78M05 = 78L05
D1 = 1N914
D2 = D3 = 1N4148 = 1N34 = 1N34A = OA76
Q1 = 2N2222 = MPS2222 = MPS3904 = BC548C
R4 = R5 = R8 = 1,2 $K\Omega$ x ¼ W y R6 = R7 = ∞ Ω

Descripción Del Circuito

En el PIC, la patilla MCLR se pone a +5V para el funcionamiento normal (no se usa aquí), a +12V para grabación, y a 0V para resetear. Realmente los +12V no "queman una EPROM" -- el voltaje superior es meramente una señal para activar el circuito interno de programación de la memoria flash. Debe ser mayor de 12 volts. La salida D0 del PC controla esta señal. No hay peligro para el chip si se aplica esta señal en un momento inadecuado.

El PIC se comunica mediante protocolo serie síncrono de dos líneas (mas masa).El pin B6 es la señal de strobel; los pulsos en este pin le indican al PIC cuando debe recibir o transmitir cada bit de datos. El Pin B7 se utiliza como entrada y salida. Cuando el PIC está recibiendo datos desde el PC, la señal SLCTIN es mantenida a nivel bajo y por lo tanto D2 no conduce por lo que D3 y R2 no se utilizan en este momento y el PIC recibe los datos mediante la señal AUTOFD.

Cuando el PIC está enviando datos, las señales SLCTIN y AUTOFD están a nivel alto, D3 no conduce y D2 y R2 proporcionan la polarización (Pull-up). La resistencia R4 mas la resistencia interna de la línea AUTOFD (dentro del puerto del PC normalmente 4.7k, auque a veces mucho menos en los nuevos puertos paralelos CMOS) proporcionan algo de Pull-up adiciona. El PC lee la información a través de la línea BUSY, que es 0.6V mayor que la salida del PIC debido al diodo D2. El puerto paralelo del PC tiene (o debería tener) entradas CMOS o Schmitt y no debería necesitar verdaderos niveles lógicos TTL.

R4 y R5 ayudan a reducir las interferencias aislando la capacidad de entrada del PIC, de modo que circule menos corriente durante transiciones bruscas. El PIC tiene entradas del tipo Schmitt, que no impiden la reducción del tiempo de subida (rise time). R8 protege la base de Q1.

El diodo D1 aumenta el voltaje de salida del 7812 en 0.6 volt, para asegurarse de que se encuentra entre los 12.0 y 14.0-v requeridos en las especificaciones del PIC.

UNIDAD Nº4

Lenguaje C

Lenguaje de programación de alto nivel desarrollado en 1972 por Dennis Mac Alistair Ritchie (9 de septiembre de 1941 - 12 de octubre de 2011) en los laboratorios Bell.

Aplicaciones

- Sistemas operativos como **iOS**(sistema operativo móvil de Apple utilizado en todos los dispositivos iPhone, iPod Touch e iPad)**, Microsoft Windows** (Software Privativo), **MacOS X y Linux** (Software Libre)
- Controladores de hardware (placas de audio, de video, de red)
- Reproductores multimedia como **Winamp**
- Aplicaciones de oficina como **Microsoft Office y LibreOffice**
- Programas de edición de imágenes digitales en forma de mapa de bits como **GIMP** (GNU Image Manipulation Program)
- Aplicaciones de compresión y descompresión de archivos como **Gzip** y **7Zip**
- Servidores web http como **Apache**
- Sistemas de gestión de bases de datos como **MySQL**
- Programas matemáticos como **Mathemática** y **Matlab**
- Aplicaciones de diseño asistido por computadora en tres dimensiones como **FreeCad**
- Motores para videos juegos tales como **Call of Duty 2, Soldier of Fortune, Quake, Doom y Half Life**
- Ensambladores
- Programación de microcontroladores
- Intérpretes de lenguajes
- Compiladores

1
1964 (emulator)
7
7-Zip

A
AAlib
AC3Filter
ActiveState Komodo
User:ADI64/DeaDBeeF
AICCU
Allegro library
Amaya (web editor)
Ambient (desktop environment)
Anaconda (installer)
Apache HTTP Server
Apcupsd
Aquamacs
Arachne (web browser)
GNU arch
Arena (web browser)
Asterisk (PBX)
Audacious (software)
Audacity
Authbind
Avahi (software)
Avidemux
Ayttm

B
Bash (Unix shell)
Beep Media Player
Berkeley Yacc
BibDesk
BitchX
BitPim
Blender (software)
Brick Engine
BRP-PACU

C
Cairo (graphics)
Carmen Toolkit
Caudium (web server)
CDex
Cdrskin
Chameleo
Chemtool
Clam AntiVirus
Claws Mail
Cloud.com
Collectd
Compiz
Concurrent Versions System
Conky (software)
Corosync (project)
CPython
CuneiForm (software)
CURL
Cygwin

D
DAP (software)
Dasher
Dcraw
Dia (software)
Dillo
Dingoo A320 SDK
Distributed Access Control System
DOSBox
Dpkg
Dynamic Computer Algebra System

E
E theorem prover
E2fsprogs
EAS3
EasyTag
Eggdrop
E cont.
Emacs
Emacspeak
Embedded GLIBC
Empathy (software)
Enhanced Machine Controller
Enterprise Volume Management System
Erwise
Eucalyptus (computing)
Evince

F
F2c
Farstream
Fast Library for Number Theory
Ffdshow
FFmpeg
FIGlet
File Roller
File System Visualizer
Filesystem in Userspace
Firefly Media Server
Flashrom
FontForge
FORM (symbolic manipulation system)
Fossil (software)
Fprint
Fre:ac
FreeMat
FreeType
Friendly interactive shell
Fuse (emulator)

G
Ganglia (software)
GD Graphics Library
Geany
Gedit
Genius (mathematics software)
Gentoo (file manager)
GeoClue
GiFT
GIMP
Git (software)
GLib
Glypha III
Gnaural
GNOME
Gnome Wave Cleaner
GnomeBaker
GNU C Library
GNU Core Utilities
GNU GRUB
GNU Multiple Precision Arithmetic Library
GNU PDF
GNU readline
GNU Scientific Library
GnuCash
Gnumeric
GNUnet
Gnuplot
GnuWin32
GOCR
GPM (software)
GPSBabel
Gpsd
GpsDrive
GPUTILS
GQview
GrafX2
GRASS GIS
Gretl
G cont.
Grip (software)
Grisbi
GROMACS
GStreamer
GThumb
GTK+
Gtk-gnutella
GtkHTML
Gtkpod
GTKWave
Gtranslator
Gummi (software)
Gzip

H
HandBrake
The Hessling Editor
Hiawatha (web server)
HMMER
HomeBank
HTML Tidy
HTMLDOC
Htop
HTTrack

I
IBM OpenDX
ImageMagick
Intelligent Input Bus
Ipchains
Irssi

J
JED (text editor)
Joe's Own Editor
Jupp (editor)
JWM

K
Kannel (telecommunications)
Kernel-based Virtual Machine
Korn shell

L
LabPlot
LADSPA
LaTeX2RTF
Libfixmath
LibHaru
Libpng
Libsndfile
Libvncserver
Libwww
Libxml2
Libxslt
LifeLines
Liferea
Lighttpd
LINA (software)
Lincity
Line Mode Browser
Linux framebuffer
Linux kernel
LIRC
Lis (linear algebra library)
LiVES
Lm sensors
LTTng
Lua (programming language)

M
MakeIndex
Mathomatic
MCSim
Md5deep
Mdadm
MDB Tools
MediaLib
MegaZeux
MEncoder
Mercurial
Mesa 3D (OpenGL)
Metacity
MiKTeX
Mined (text editor)
MinGW
Minicom
Mintty
Miredo
MLDonkey
Mobile Web Server (Symbian OS)
Mod openpgp
Mondo Rescue
Moonlight (runtime)
MP3Gain
Mp3splt
MPIR (mathematics software)
MPlayer
Mrxvt
Music on Console
Music Player Daemon
Mutter (window manager)

N
GNU nano
Nautilus (file manager)
NaviServer
Ncurses
Ne (text editor)
NEdit
Netpbm
Netsniff-ng
Network Audio System
Network UPS Tools
User:Neustradamus/CTorrent
Nginx
Ngrep
Notepad2
Nouveau (software)
NSPluginWrapper
Ntfsprogs
Nullsoft Scriptable Install System
NuSMV

O
Openbox
OpenLDAP
OpenNebula
Oroboros (window manager)
ORX
OsFree

P
Packetsquare
Palimpsest Disk Utility
Pango
Panorama Tools
P cont.
PARI/GP
Parrot virtual machine
Pcap
PCMan File Manager
Perl
Pidgin (software)
Pigment (software)
Ploticus
Pngcrush
PortAudio
PostgreSQL
Potrace
Praat
Premake
Privoxy
PSPP
PulseAudio
Puma.NET
PuTTY

Q
Qmail
Qodem
QuickSynergy

R
Radvd
RasMol
ReactOS
RecordMyDesktop
Remmina
Rhythmbox
Rlab
ROX Desktop
RRDtool
Ruby MRI

S
SeaBIOS
Seahorse (software)
Sge2d
Shoes (GUI toolkit)
SK1 (program)
Skencil
SmallBASIC
Smartmontools
SnapPea
Snappy
Sodipodi
SoftGenLock
SoX
SPICE
Sqsh
Stratagus
Streamripper
Apache Subversion
Sylpheed

T
Tcpdump
Tcsh
Tesseract (software)
T cont.
Thoggen
Thttpd
TigerVNC
TightVNC
TiLP
TiMidity++
TinTin++
Tor (anonymity network)
Totem (software)
Transmission (BitTorrent client)
Tsclient
Tux Paint
Tvtime

U
UFO: Alien Invasion
UFRaw
UltraDefrag
UltraVNC
UMFPACK
Unbound (DNS Server)
Unified Code Count (UCC)
Unladen Swallow
Usplash
Utah GLX
Util-linux
Uzbl

V
V9fs
Varnish (software)
Vim (text editor)
VirtualDubMod

W
Wayland (display server protocol)
Website Meta Language
WeeChat
Wine (software)
Wireshark

X
X Neural Switcher
X2vnc
X2x
Xarchiver
Xdebug
XEmacs
Xine
XMD
XMMS
Xterm
XVidCap
Xxxterm
XZ Utils
Xzgv

Y
Yafc
YXORP

Z
ZBar
Zgv
Zlib
ZSNES

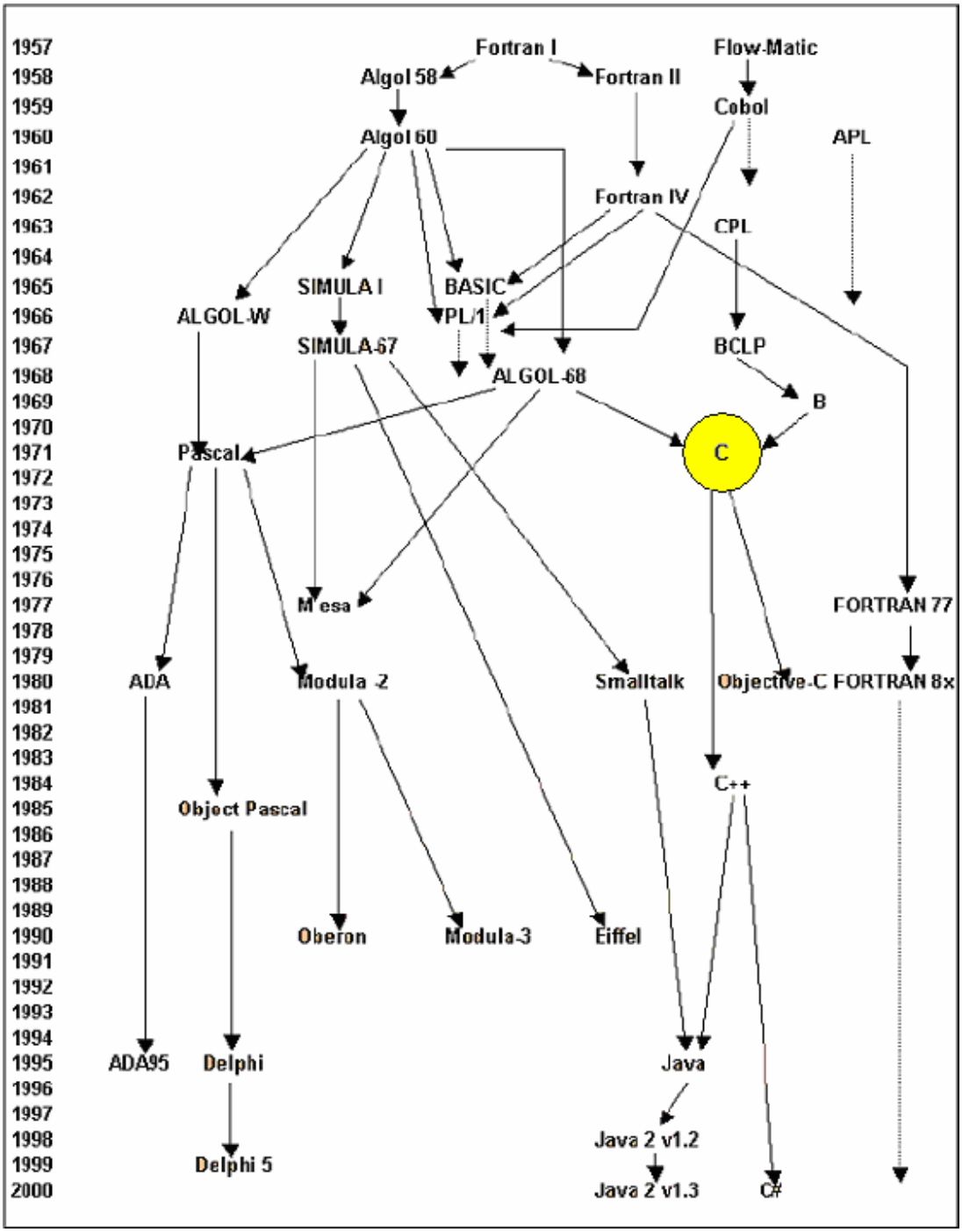
Popularidad Mundial

Octubre del 2011

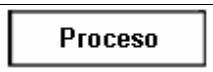
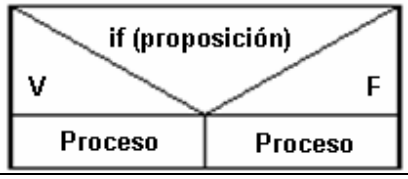
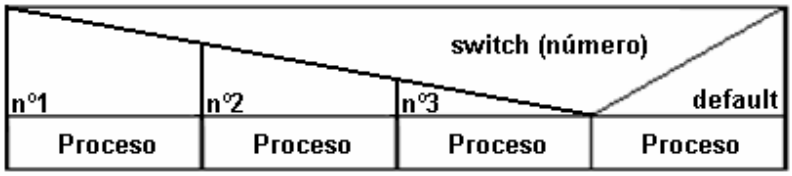
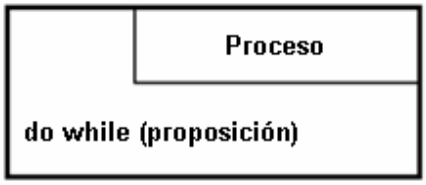
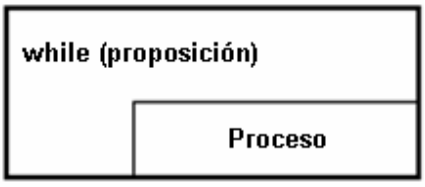
Tendencia	Lenguaje De Programación
=	Java
=	C
=	C++
=	PHP
↑	C#
↑↑	Objective-C
↓↓	(Visual) Basic
↓	Python
=	Perl
↑	JavaScript
↓	Ruby
=	Delphi/Object Pascal
=	Lisp
=	Transact-SQL
↑↑↑↑↑↑↑	PL/SQL
↑↑↑↑↑↑↑	Lua
↓	RPG (OS/400)
↓↓↓	Pascal
=	Assembly
↓↓↓	Ada

Fuente: http://www.tiobe.com

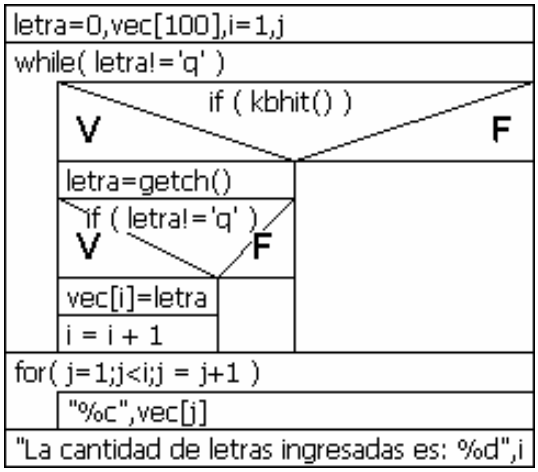
Lenguajes Antecesores Y Sucesores



Representación Del Programa En Diagramas De Chapin

SIMBOLOS	
Gráfico	**Representación**
	**Proceso**
	**if (si)**
	**switch (conmutador)**
	**do while (hacer mientras)**
	**while (mientras)**

Ejemplo



Codificación Del Programa En C

Inclusión de los encabezamientos

```
#include <stdio.h>
#include <stdlib.h>
```

Declaración de variables globales

[tipo de variable] [nombre de la variable;] [//comentario]

```
int var1=1;      // declaro var1 como una variable global de tipo entero y la inicializo en 1
float var2=2.5; // declaro var2 como una variable global de tipo real y la inicializo en 2,5
```

Declaración de funciones

```
int sumar(int x, int y)
{ return (x+y); }

void mostrar(int z)
{ printf("resultado: %d",z); }
```

Inicio del programa

```
int main (void)
{
```

Declaración de variables locales

[tipo de variable] [nombre de la variable;] [//comentario]

```
int var3=3;      // declaro var3 como una variable local de tipo real y la inicializo en 3
float var4=4.5; // declaro var4 como una variable local de tipo real y la inicializo en 4,5
int resultado;   // declaro resultado como una variable local de tipo real y no la inicializo
```

Cuerpo del programa

Sentencias de control del flujo (if, switch case, while, do while y for) y Llamadas a Funciones

```
system("cls");                    // borra la pantalla
printf("Primer Programa en C "); // imprime un mensaje en la pantalla
printf("var1: %d ",var1);         // imprime el contenido de var1 en la pantalla
printf("var2: %f ",var2);         // imprime el contenido de var2 en la pantalla
printf("var3: %d ",var3);         // imprime el contenido de var3 en la pantalla
printf("var4: %f ",var4);         // imprime el contenido de var4 en la pantalla
resultado=sumar(var1,var3);      // llamo a la función sumar
mostrar(resultado);               // llamo a la función mostrar
system("pause");                  // se queda pausado hasta que se presione alguna tecla
```

Retorno del main

```
return 0;
```

Fin del programa

```
}
```

Compiladores De C

- **Bloodshed Dev-C++** (Software Libre)
- **Code::Blocks** (Software Libre)
- **Borland Turbo C** (Software Privativo)

Tipos De Datos

Tipo De Dato En C	Código Que Utiliza	Tamaño En Bytes	Tipo De Números Que Almacena	Rango De Valores				Especificador De Formato
				signed		unsigned		
				mínimo	máximo	mínimo	máximo	
char	ASCII y ASCII extendido	1	enteros	-128	127	0	255	%c
int	cá2	2	enteros	-32.768	32.767	0	65.535	%i(signed) %u(unsigned)
short int	cá2	2	enteros	-32.768	32.767	0	65.535	%d
long int	cá2	4	enteros	-2.147.483.648	2.147.483.647	0	4.294.967.295	%D(signed) %U(unsigned)
float	coma flotante	4	reales	$3{,}4^{-38}$	$3{,}4^{38}$	No Existe		%f
double	coma flotante	8	reales	$1{,}7^{-308}$	$1{,}7^{308}$	No Existe		%lf
long double	coma flotante	10	reales	$3{,}4^{-4932}$	$1{,}1^{4932}$	No Existe		%LF

Declaración

<tipo de dato> <nombre del dato 1>, <nombre del dato 2>, <nombre del dato 3>;

Sizeof

Operador que devuelve la longitud en bytes (valor entero), de la variable o del especificador del tipo.

Ejemplo

Código Fuente

```
#include<stdio.h>
#include<stdlib.h >

int main (void)
{
 float f; //declaración del dato "f" del tipo float
 printf("%i ",sizeof f);
 printf("%i ",sizeof int);
 system("pause");
}
```

Salida Por Pantalla

```
4 2(*)
```

(*) Dependiendo el compilador puede ser 2 ó 4

Operadores

Aritméticos	**+**	suma
	-	resta
	*****	producto
	/	división
	%	módulo (resto de una división entera)
	++	incremento
	--	decremento
Bit a Bit	**<<**	desplazamiento a la izquierda
	>>	desplazamiento a la derecha
	&	and de bit
	|	or de bit
	^	xor de bit
	~	complemento a uno
Asignación	**=**	asignación simple
	+=	asignación con suma
	-=	asignación con resta
	***=**	asignación con producto
	/=	asignación con cociente
	%=	asignación con módulo (resto de una división entera)
	<<=	asignación con desplazamiento a la izquierda
	>>=	asignación con desplazamiento a la derecha
	&=	asignación con and
	|=	asignación con or
	^=	asignación con xor

Equivalencias De Sintaxis

Var=Var+Num;	Var+=Num;
Var=Var-Num;	Var-=Num;
Var=Var*Num;	Var*=Num;
Var=Var/Num;	Var/=Num;
Var=Var%2;	Var%=2;
Var=Var+Num;	Var+=Num;
Var=Var+1;	Var++; ó ++Var;
Var=Var-1;	Var--; ó –Var;
Var2=Var1; Var1=Var1+1;	Var2=Var1++;
Var1=Var1+1; Var2=Var1;	Var2=++Var1;
Var2=Var1; Var1=Var1-1;	Var2=Var1--;
Var1=Var1-1; Var2=Var1;	Var2=--Var1;

Operadores

Relacionales	**>**	mayor que
	>=	mayor o igual que
	<	menor que
	<=	menor o igual que
	==	igual que
	!=	distinto de
Lógicos	**!**	not
	&&	and
	||	or
	^	xor

Control Del Flujo

Estructuras Condicionales

If	Switch-case
if(proposición) { .... } else { .... }	switch (variable) { case 1: .... break; case 2: .... break; default: .... break; }

Estructuras Iterativas

For	While	Do While
for(instrucción inicial;proposición;instrucción de fin de bloque) { .... }	while (proposición) { .... }	do { .... } while (proposición);

Vectores

Variables comunes que llevan el mismo nombre y se diferencian por un índice.

Declaración

<tipo de dato> <nombre del vector>[<cantidad de elementos>];

Ejemplo

Código Fuente

```
#include<stdio.h>
#include<stdlib.h >

int main (void)
{
 int vector[3]={4,5,6}; //El vector tendrá 3 elementos (el elemento 0, 1 y 2)
 printf("%i %i %i", vector[0], vector[1], vector[2]);
 system("pause");
}
```

Salida Por Pantalla

```
4 5 6
```

Índice

Identifica un elemento en particular (valor entero) dentro del vector.

Ejemplo

Código Fuente

```
#include<stdio.h>
#include<stdlib.h >

int main (void)
{
 int vector[3]; //El vector tendrá 3 elementos (el elemento 0, 1 y 2)
 vector[0]=1; //
 vector[1]=2;
 vector[2]=3;
 printf("%i %i %i", vector[0], vector[1], vector[2]);
 system("pause");
}
```

Salida Por Pantalla

```
1 2 3
```

Ejemplo

Código Fuente

```
#include<stdio.h>
#include<stdlib.h >

int main (void)
{
 int vector[3];
 int i; //Esta variable será el índice del vector

 for(i=0;i<3;i++)
 {
  printf("Ingrese el contenido de la componente %i del vector: ",i+1);
  scanf("%i",&vector[i]);
 }

 for(i=0;i<3;i++)
 {
  printf("El contenido de la componente %i del vector es: %i \n",i+1,vector[i]);
 }
 system("pause");
}
```

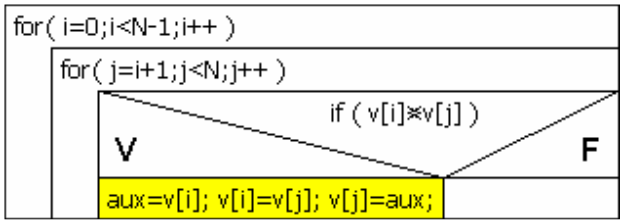
Salida Por Pantalla

```
Ingrese el contenido de la componente 1 del vector: 9
Ingrese el contenido de la componente 2 del vector: 8
Ingrese el contenido de la componente 3 del vector: 7
El contenido de la componente 1 del vector es: 9
El contenido de la componente 2 del vector es: 8
El contenido de la componente 3 del vector es: 7
```

Métodos De Ordenamiento De Vectores

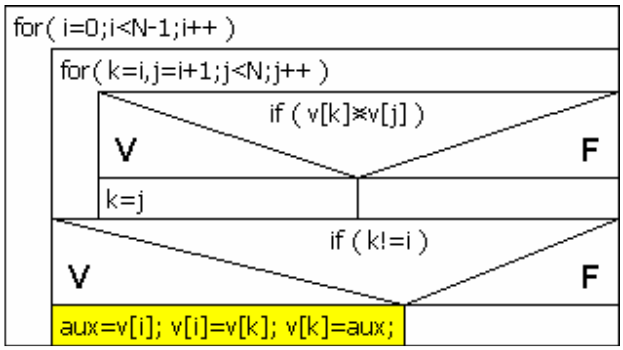
N = cantidad de elementos del vector

Pivote



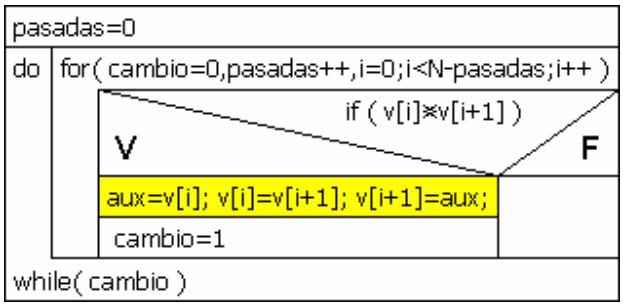
Cantidad de comparaciones necesarias para el ordenamiento (no depende del grado de desorden del vector)	$C = \frac{N^2 - N}{2}$

Selección



Cantidad de comparaciones necesarias para el ordenamiento (no depende del grado de desorden del vector)	$C = \frac{N^2 - N}{2}$

Burbuja



Cantidad de comparaciones necesarias para el ordenamiento (depende del grado de desorden del vector)	$N - 1 \leq C \leq \frac{N^2 - N}{2}$

ordenamiento ascendente	Si ⋇ es >
ordenamiento descendente	Si ⋇ es <

Cadena De Caracteres (String)

Vector de caracteres cuyo último elemento es el carácter nulo ('\0'), el cual indica el final de la cadena. El especificador de formato para los String es el "%s".

Ejemplo

Código Fuente

```
#include<stdio.h>
#include<stdlib.h >

int main (void)
{
 char vector[4]={'a','b','c','\0'};
// Es equivalente a poner "char vector[4]={97,98,99,0};"
// Es equivalente a poner "char vector[4]="abc";"
 printf("%s", vector);
}
```

Salida Por Pantalla

```
abc
```

Ejemplo

Código Fuente

```
#include<stdio.h>
#include<stdlib.h >

int main (void)
{
 char palabra[11];
 //La palabra podra tener como máximo 10 letras ya que reservamos 1 espacio para el carácter nulo ( '\0')
 int i;
 printf("Ingrese una palabra de hasta 10 letras: ");
 scanf("%s",palabra); //No se usa el "&"

 for(i=0;i<11;i++)
 {
  printf("El contenido de la posición %i del vector palabra es: '%c'\n",i+1,palabra[i]);
 }
}
```

Salida Por Pantalla

```
Ingrese una palabra de hasta 10 letras: holamundo
El contenido de la posición 1 del vector palabra es: 'h'
El contenido de la posición 2 del vector palabra es: 'o'
El contenido de la posición 3 del vector palabra es: 'l'
El contenido de la posición 4 del vector palabra es: 'a'
El contenido de la posición 5 del vector palabra es: 'm'
El contenido de la posición 6 del vector palabra es: 'u'
El contenido de la posición 7 del vector palabra es: 'n'
El contenido de la posición 8 del vector palabra es: 'd'
El contenido de la posición 9 del vector palabra es: 'o'
El contenido de la posición 10 del vector palabra es: ' '
El contenido de la posición 11 del vector palabra es: '<basura>'
```